

## PERTEMUAN 11

### IMPLEMENTASI DIAGRAM UML

#### A. TUJUAN PMBELAJARAN

Pada pertemuan ini dijelaskan tentang tatacara bagaimana cara membuat diagram UML, serta mengetahui fungsi kegunaan pembuatan Use Case Diagram, Class Diagram Serta Object Diagram.

#### B. URAIAN MATERI

##### 1. Use Case Diagram

Ketika ingin membuat sebuah aplikasi, diperlukan sebuah rancangan, biasanya rancangan tersebut digunakan untuk skenario menjalankan suatu sistem. Tujuan pembuatan skenario ini untuk memberikan suatu gambaran hal-hal apa saja yang akan dibutuhkan pada aplikasi ketika aplikasi tersebut dibuat, selain itu juga sebagai acuan desain ketika kita membuat aplikasi, lalu berfungsi juga untuk membatasi beberapa hal-hal persyaratan yang dapat divalidasi ketika aplikasi dibuat suatu rancangan yang disebutkan sebelumnya dapat kita sebut sebagai Use Case Diagram. Use Case itu sendiri adalah suatu fungsi yang berisi penjelasan-penjelasan atau deskripsi yang memudahkan suatu pengguna atau *user* ketika ingin mempelajari suatu fungsi pada sistem.

Dengan metedologi tradisional, setiap system diuraikan menjadi satu set subsistem, yang tiap setiap gilirannya diuraikan menjadi subsistem lebih lanjut, dan seterusnya. Hal ini berlangsung sampai tidak ada proses penguraian yang masuk akal, dan hal itu sering terjadi dan membutuhkan banyaknya halaman diagram agar terkait satu sama lain. Pada kasus yang berfokus pada satu proses bisnis, biasanya perancangan model sistem akan dibuat menjadi lebih sederhana.

Untuk memahami lebih lanjut tentang penggunaan Use Case Diagram, mari masuk lebih bagian-bagian dari Use Case Diagram itu sendiri.

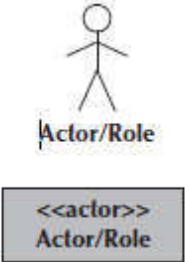
### a. Elemen-Elemen Pada Use Case Diagram


Pada penjelasan sebelumnya dijelaskan bahwa pada saat berencana membuat suatu program diperlukan yang namanya suatu rancangan. Dengan rancangan yang dibuat dapat mengidentifikasi sebuah proses yang terjadi pada suatu sistem serta menguraikan setiap proses secara detail. Diperlukan juga sebuah analisa ketika merancang sebuah diagram, yang nantinya dapat mempermudah user untuk mengetahui tiap-tiap fungsi yang terjadi disebuah sistem.

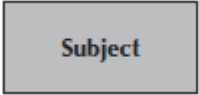

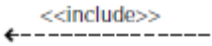
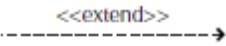
Elemen-Elemen dari Use Case Diagram yaitu ada yang Namanya actor, use cases, batasan subjek, dan relasi atau hubungan antara actor dan use case. Pada relasi ada yang Namanya relasi *association*, *include*, *extend*, dan relasi *generalization*.


Actor bukan hanya berfungsi sebagai pengguna, tetapi merupakan peran yang dapat memainkan fungsi pengguna ketika berinteraksi pada sebuah sistem. Actor juga dapat mewakili sistem lain dimana sistem saat ini berinteraksi. Secara opsional aktor juga dapat mewakili didalam suatu sistem yang digambarkan bentuk persegi panjang yang berisikan “<<actor>>” dan nama dari sistem. Pada dasarnya Actor merupakan unsur penting dalam ruang lingkup suatu sistem.

Berikut ini akan ditampilkan elemen apa saja yang ada pada Use Case Diagram.

<p><b>Actor:</b></p> <ul style="list-style-type: none"> <li>▪ Actor bisa saja user atau suatu fungsi sistem yang menggunakan fungsi atau manfaat yang berada didalam maupun diluar ruang lingkup pada suatu subjek.</li> <li>▪ Jika actor yang memfungsikan suatu sistem adalah manusia maka dilambangkan sebagai character gambar manusia seperti gambar disamping, jika</li> </ul>	
--	---

<p>aktor yang terkait bukan manusia, maka akan dilambangkan sebagai persegi panjang.</p> <ul style="list-style-type: none"> <li>▪ Tiap aktor dilabeli sebuah peran yang dicatumkan dengan sebuah nama.</li> <li>▪ Aktor dapat direlasi kan dengan actor lain melalui fungsi <i>specialization/super class association</i>, yang digambarkan dengan sebuah anak panah.</li> <li>▪ Actor dapat diletakan diruang lingkup suatu sistem.</li> </ul>	
<p><b>Use Case:</b></p> <ul style="list-style-type: none"> <li>▪ Merepresentasikan bagian-bagian pada suatu sistem yang fungsional.</li> <li>▪ Dapat merelasikan fungsi extend dengan use case lain.</li> <li>▪ Dapat merelasikan fungsi include dengan use case lain.</li> <li>▪ Dapat diletakan didalam ruang lingkup suatu sistem.</li> </ul>	

<ul style="list-style-type: none"> <li>Dapat namakan dengan deskripsi kata kerja – frase kata benda.</li> </ul>	
<p><b>Subjek:</b></p> <ul style="list-style-type: none"> <li>Penamaan subjek dapat diisi didalam ataupun diatas subjek.</li> <li>Merepresentasikan sebuah ruang lingkup pada suatu sistem yang didalamnya terdapat fungsi proses.</li> </ul>	
<p><b>Relasi association:</b></p> <ul style="list-style-type: none"> <li>Berfungsi untuk menghubungkan antara actor dan usecase agar saling berinteraksi.</li> </ul>	
<p><b>Relasi include:</b></p> <ul style="list-style-type: none"> <li>Merepresentasikan fungsi relasi include dari antara 1 use case ke use case lainnya.</li> <li>Gambar panah digambarkan dari use case utama menuju use case yang aktif.</li> </ul>	
<p><b>Relasi extend:</b></p> <ul style="list-style-type: none"> <li>Merepresentasikan fungsi relasi extend antara 1 use case ke use case lainnya.</li> <li>Gambar panah digambarkan dari</li> </ul>	

use case ke use case utama.	
<b>Relasi generalization:</b> <ul style="list-style-type: none"> <li>▪ Mempresentasikan use case khusus ke arah yang umum.</li> <li>▪ Gambar panah digambarkan dari use case khusus ke use case utama.</li> </ul>	

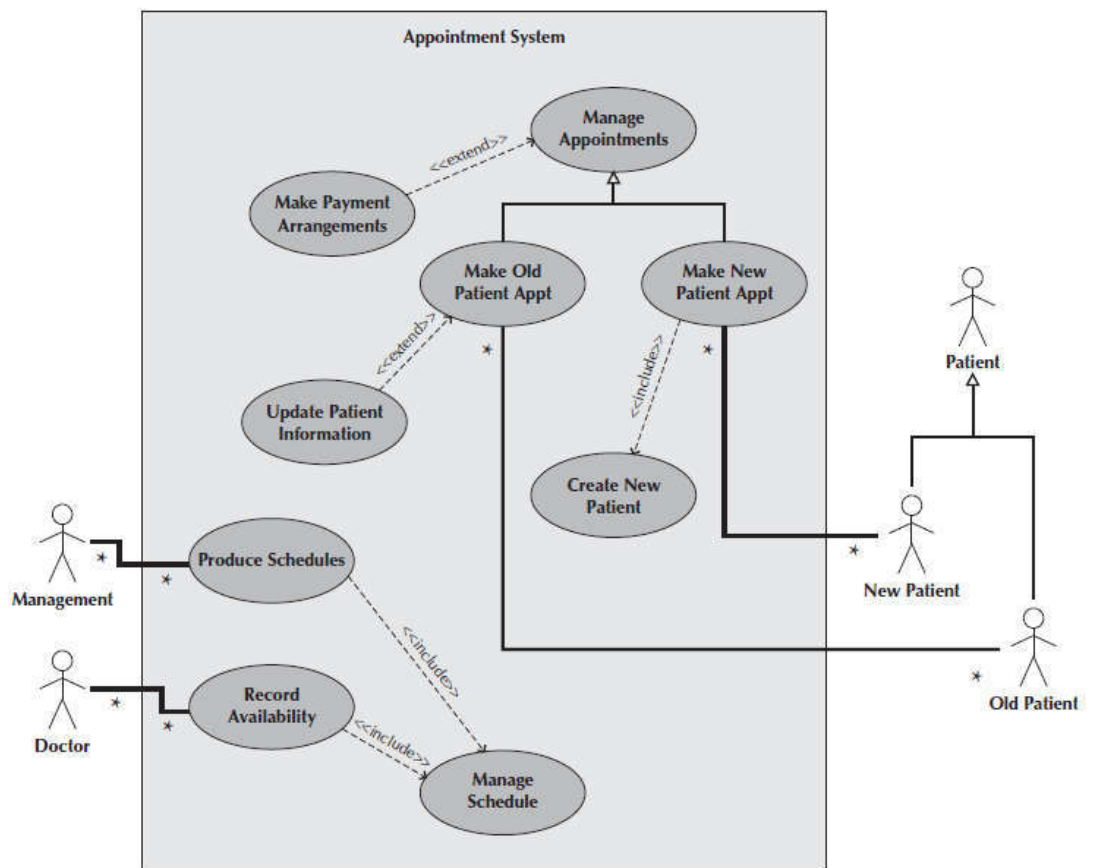
Tabel 11. 1 Tabel Elemen pada Use Case Diagram

Selanjutnya agar mahasiswa lebih memahami dengan tata cara penggunaan use case maka akan dijelaskan lebih lanjut lagi melalui study kasus.

#### b. Studi Kasus Pada Rumah sakit

Sebelumnya sudah dijelaskan elemen-elemen apa saja yang menjadi bagian penggunaan pada Use Case Diagram ketika merancang suatu sistem oleh karena itu akan dijelaskan tata cara penggunaan use case diagram dengan studi kasus langsung.

Pada Studi kasus ini dikondisikan terkadang suatu aktor dapat memainkan peran khusus dari tipe aktor secara umum. Sebagai contoh, kadang kala pasien baru berinteraksi dengan sistem dengan cara yang agak berbeda dari pasien pada umumnya. Dalam hal ini, aktor khusus(pasienbaru) dapat ditempatkan pada model, yang ditampilkan menggunakan garis dengan segitiga berlubang pada ujungnya.



Gambar 11. 1 Gambar use case pada sistem rumah sakit

Nah Pada Studi kasis ini ada 1 aktor yang mempunyai hak khusus langsung tanpa harus melakukan beberapa prosedur khusus yang dibiasa dilakukan pada aktor umumnya, yaitu pasien tua. Pasien tua ini bisa langsung masuk ruang lingkup sistem tanpa harus memenuhi beberapa aktivitas yang biasa dilakukan pasien lain pada umumnya. Dikarenakan pasien tua memiliki hak khusus yang berbeda pada pasien lainnya. Itulah kenapa adanya suatu aktor dapat kondisikan mempunyai hak khusus dari aktor lain pada umumnya yang mengikuti aturan pada sistem dikarenakan sebuah aktor dapat diperlakukan secara khusus.

Ruang lingkup sistem use case diapit oleh batas subjek, yang dimana sebuah kotak didefinisikan ruang lingkup sistem dan jelas digambarkan didiagram itu. Satu keputusan yang sulit dibuat dikarenakan tempat menggambar pada batas subjek. Batas Subjek ini dapat digunakan untuk memisahkan sistem perangkat lunak dari ruang lingkup sistem, subsistem dari subsistem lain pada perangkat lunak sistem, atau proses individu dalam

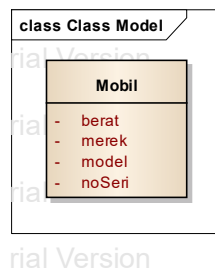
sistem perangkat lunak. Ini juga dapat digunakan sebagai pemisah sistem informasi, termasuk perangkat lunak dan actor dalam, dari ruang lingkup sistem. Nama Subjek dapat muncul di dalam atau di atas kotak. Subjek batas ditarik berdasarkan ruang lingkup sistem. Dalam sistem penunjukan, diasumsikan bahwa para actor manajemen dan doctor, berada pada ruang lingkup pada suatu sistem, dikarenakan merekalah yang menggunakan sistem tersebut. Sistem dapat memasukan resepsionis sebagai aktor. Namun pada kasus ini, diasumsikan bahwa resepsionis, adalah aktor dalam yang merupakan bagian dari kasus penggunaan Kelola janji temu yang berinteraksi dengan aktor pasien. Oleh karena itu resepsionis tidak digambarkan pada diagram diatas.

## 2. Class Diagram

Class itu sendiri yaitu sekumpulan obyek yang dimana setiap obyeknya memiliki atribut dan operasi. Jadi Class Diagram itu sendiri suatu diagram yang dimana didalamnya terbuat dari serangkaian objek, dimana masing-masing objek mempunyai atribut dan operasi. Oleh karena itu akan dijelaskan elemen-elemen yang berada pada tiap-tiap 1 class pada suatu class diagram.

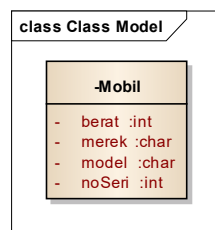
### a. Attribute.

Attribute merupakan elemen yang ada pada suatu class. Attribute menggambarkan batas nilai yang mungkin ada pada obyek dalam suatu class. Satu class bisa saja mempunyai kosong atau lebih dari 0 atribut. Secara kovenensi, jika nama attribute terdiri dari sebuah suku kata, maka dideskripsikan dengan huruf kecil. Tetapi jika nama attribute mempunyai lebih dari sebuah suku kata maka semua suku kata tergabung dengan suku kata awal memakai huruf kecil dan awal suku kata berikutnya memakai huruf besar. Berikut ini contoh dari atribut yang berada pada suatu mobil.



Gambar 11. 2 Gambar class mobil

Jadi pada suatu mobil mempunyai elemen-elemen atribut pada benda tersebut, pada suatu Class tiap-tiap atribut memiliki type data masing-masing sesuai keinginan siperancang suatu aplikasi ketika ingin mengidentifikasi suatu atribut yang berada pada class diagram.



Gambar 11. 3 Class dan type datanya.

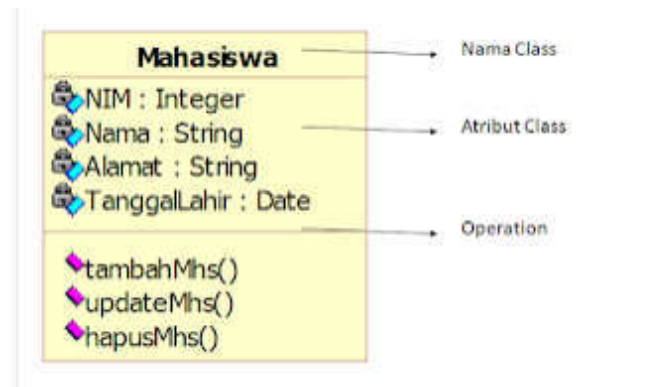
Jadi Berat pada class disini diidentifikasi sebagai *integer*, lalu untuk merek sebagai character atau deskripsi, lalu model juga sebagai *character* atau deskripsi, dan terakhir untuk noSeri saya klasifikasikan sebagai *integer*, noSeri ini, sebagai primary key, atau penanda pada suatu mobil memiliki nomor kode produksi unik tersendiri yang berbeda pada setiap mobilnya, meskipun dengan merek model yang sama, tetapi setiap mobil kendaraan yang diproduksi suatu perusahaan memiliki noSeri pembuatan yang berbeda. Hal inilah yang nantinya agar dapat mengidentifikasi data kedalam database secara mudah.

#### b. Operation

Operation adalah sesuatu yang hanya dikerjakan pada sebuah class yang hanya dapat dikerjakan oleh class itu sendiri. Setiap class mempunyai fungsi masing-masing ketika melakukan aktivitas. Akan dijelaskan



seperti contoh dibawah ini.



Gambar 11. 4 Class diagram yang memiliki fungsi operation

Yang berada diposisi kotak dibawah adalah sebuah operasi, fungsi ini sangat membantu mendeskripsikan suatu fungsi pada tiap-tiap kelas yang pada nantinya akan memudahkan seseorang ketika merancang suatu aplikasi, dengan deskripsi ini dapat mengetahui bawah fungsi apa saja yang nantinya akan dimasukan pada suatu class, agar memunuhi seluruh kebutuhan yang ada pada suatu class tersebut.

#### c. Penyederhanaan Class Diagram

Saat Class Diagram terisi penuh dengan semua class dan relasi dengan dunia nyata sistem, class diagram bisa menjadi sulit untuk diartikan atau bisa menjadi lebih kompleks dimengerti oleh orang awam. Jika ini terjadi, terkadang diagram perlu disederhanakan. Namung tergantung pada jumlah asosiasi yang terhubung ke kelas abstrak, jika banyaknya maka akan sulit dipahami diagram yang kita buat.

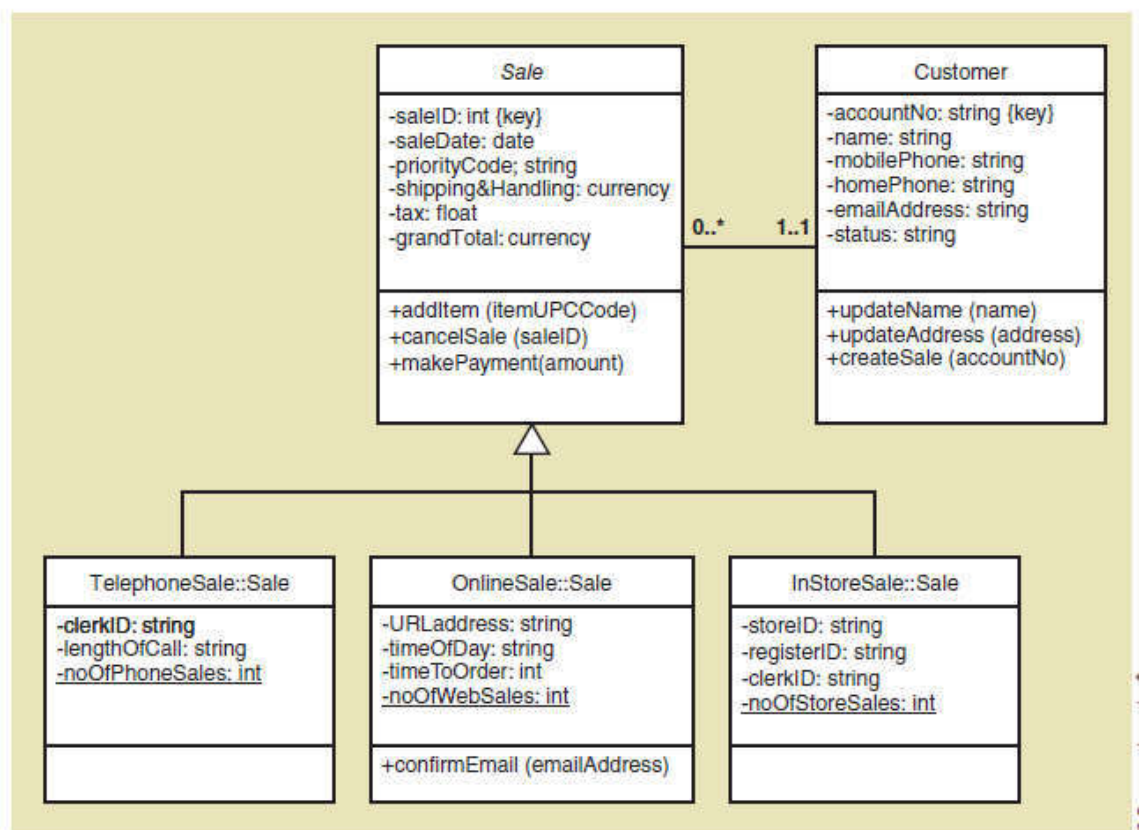
Cara untuk menderhanakan class diagram adalah dengan menggunakan tampilan view. Tampilan awalnya dikembangkan dengan sistem manajemen database relasional untuk ditampilkan saja yang berada pada database. Dalam hal ini, view akan berguna pada class diagram yang menampilkan relasi antar class yang relevan. Pada sudut pandang lain akan menunjukan relasi-relasi tertentu, seperti agregasi, asosiasi, atau generalisasi. Setiap kelas, misalnya, hanya menampilkan nama class dan atribut, atau nama class dan operasi.

Care selanjutnya untuk menderhanakan class diagram yaitu dengan menggunakan fungsi *package*. Untuk membuat diagram lebih mudah dibaca

dan menyimpan model-model. Dengan menggunakan *package* tingkat kompleksitas pada suatu clas-class akan dikelompokkan menjadi sebuah *package*. *Package* adalah konstruksi umum yang dapat diterapkan ke salah satu elemen dalam model UML.

d. Studi Kasus Class Diagram Pada Sistem Perbelanjaan Online

Berikut ini contoh studi kasus Bentuk class diagram, pada sistem perbelanjaan online dan beserta deskripsi penjelasannya.



Gambar 11. 5 Contoh Class Diagram pemesanan belanja online.

Bisa dilihat pada kolom Class Sale terdapat atribut *saleID*, *saleDate*, *priorityCode*, *shipping&Handling*, *tax*, *grandTotal*. Yang mempunyai operasi, *add item*, *cancelSale*, *makePayment*. Pada Class *costumer* terdapat atribut *account*, *name*, *mobilePhone*, *homePhone*, *emailAddress*, dan *status* dan memiliki operasi *update*, *updateAddress*, *createSale*. Pada *TelephoneSale::Sale* terdapat atribut, *clerkID*, *lengthOfCall*, *noOfPhoneSales*. Pada class *OnlineSale::Sale* terdapat atribut *URLaddress*,

*timeOfDay*, *timeToOrder*, *noOfWebSales* dan memiliki fungsi operasi *confirmEmail*. Pada class *InStoreSale* terdapat atribut *storeId*, *registerID*, *clerkID*, *noOfStoreSales*.

Dari kesimpulan diatas bahwa tidak selalu setiap class mempunyai sebuah operasi. Dikarenakan Tiap class mempunyai karakteristik masing-masing sesuai kebutuhan sang pembuat aplikasi ketika merancang suatu aplikasi, disini sangat diperlukan analisa yang cukup baik untuk mengidentifikasi fungsi-fungsi tiap class, agar aplikasi nantinya dapat berjalan sesuai dengan permintaan user.

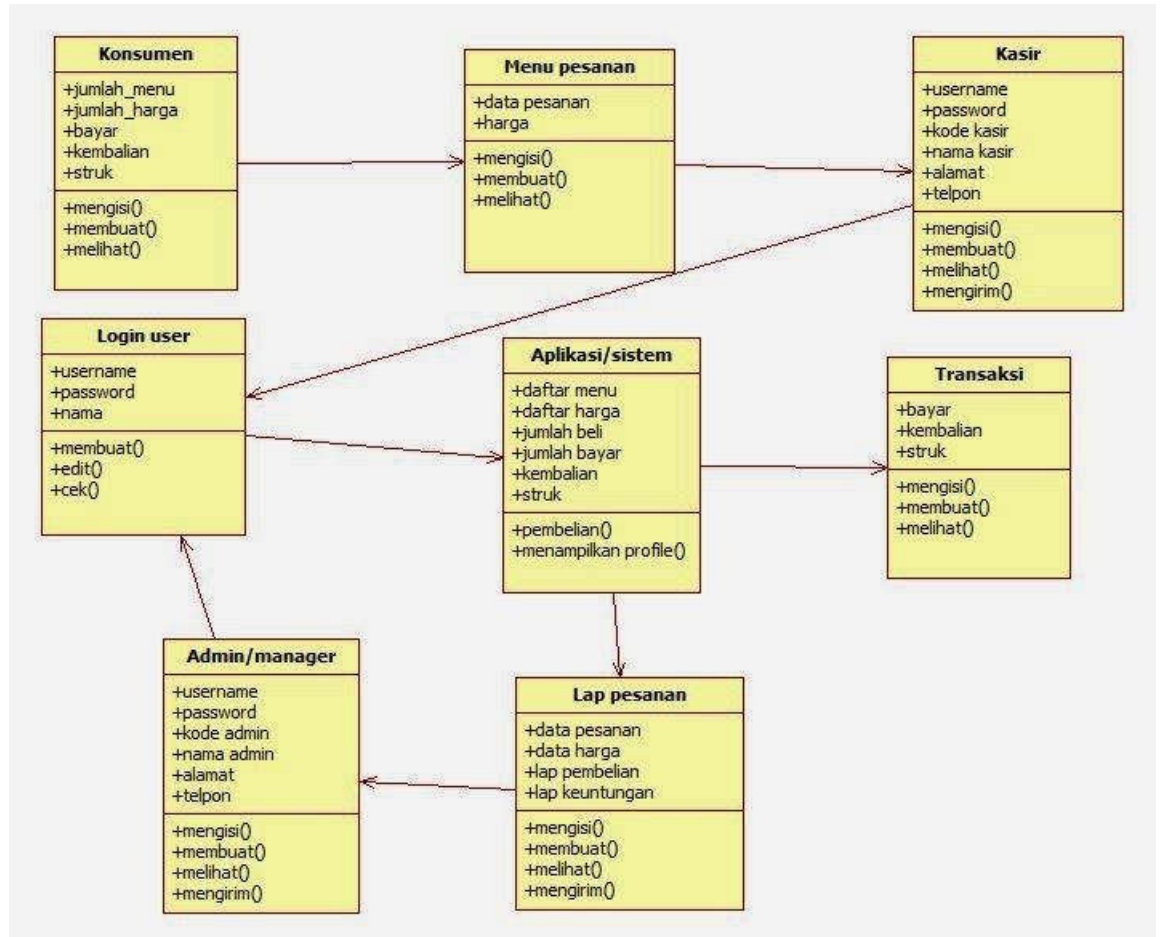
### 3. Object Diagram

Pembahasan selanjutnya adalah Object Diagram, Object Diagram itu sendiri sebenarnya berasal dari class diagram, hanya saja pada class diagram hanya dijelaskan class, atribut, dan nama operasi secara abstrak, sedangkan pada object diagram menggambarkan obyek-obyek dengan jelas, Dengan menjelaskan dan mendeskripsikan tiap-tiap fungsi pada suatu atribut agar membuat para user dengan mudah memahami suatu sistem aplikasi yang sedang kita buat. Tujuan dibuatnya Object Diagram agar user dapat membandingkan perbedaan yang mewakili class diagram dengan object diagram atau contoh aktivitas yang berada pada setiap elemen-elemen atribut pada Class Diagram. Secara ringkas tujuan dibuatnya Object diagram adalah menunjukkan relasi objek dari class ke class yang lain pada suatu sistem, memahami perilaku objek dan hubungan mereka antara 1 class dengan class yang lain.

Meskipun diagram kelas diperlukan untuk mendokumentasikan struktur kelas, sedetik saja. Jenis diagram struktur statis, disebut diagram objek karena berguna untuk mengungkapkan tambahan informasi. Object Diagram pada dasarnya adalah contoh dari semua atau sebagian Class Diagram. Instansiasi berarti berarti membuat instance kelas dengan himpunan yang sesuai nilai atribut. Diagram objek bisa sangat berguna saat mencoba mengungkap detail class. Singkatnya pada object diagram lebih konkret daripada Class diagram yang pengungkapannya masih terbilang abstrak.

Berikut ini salah 1 contoh object diagram pada sistem yang ada pada pemesanan makan di restoran.

a. Studi Kasus Object Diagram Pada Sistem Pemesanan Makanan Di Restoran



Gambar 11. 6 Object Diagram pemesanan makan di restoran

Jadi sudah terlihat jelas perbedaan dari Class Diagram dan Object Diagram, Jika pada Class Diagram kita hanya menjelaskan type elemen pada suatu atribut yang berada didalam class serta perelasannya dengan class yang lain, sedangkan pada Object diagram kita dapat melihat jelas suatu fungsi pada suatu atribut pada tiap classnya serta relasinya yang dapat kita pahami dengan mudah.

Pada Object Diagram, kita dapat menerjemahkan informasi-informasi yang ada pada suatu class, tanpa melihat detail secara abstraksi. Semua aktifitas terdeskripsi sangat jelas beserta operation pada tiap-tiap class.

#### b. Tujuan Object Diagram

Tujuan Object diagram mirip dengan class diagram. Bedannya, class diagram mewakili model abstrak yang terdiri atas class-class dan hubungannya. Sedangkan Object Diagram merupakan contoh langsung dari Class Diagram. Tujuannya adalah menangkap pandangan statis pada sebuah sistem.

Secara ringkas, tujuan object diagram yaitu Untuk *forward* dan *reverse engineering*, Menunjukkan relasi objek dari suatu sistem, menunjukkan pandangan statis dari suatu interaksi, serta memahami perilaku objek dan hubungan mereka dari perspektif praktis.

#### c. Penggunaan Object Diagram

Object diagram sangat berguna dalam menampilkan sampel-sampel obyek yang berhubungan antara satu dengan lainnya. Dalam banyak contoh, rangkaian yang tepat dapat dideskripsikan secara tepat menggunakan diagram kelas, tetapi struktur tersebut masih bersifat abstrak dan sulit untuk dipahami. Pada keadaan seperti ini membuat contoh dengan obyek diagram akan mempermudah pengerjaan. Untuk memodelkan sebuah struktur obyek bisa dilakukan dengan langkah-langkah sebagai berikut :

- 1) Identifikasikan mekanisme yang ingin dimodelkan. Sebuah mekanisme mewakili beberapa fungsi atau perilaku dari Sebagian sistem yang akan dimodelkan sebagai hasil dari interaksi dari class, interface dan hal-hal yang lain.
- 2) Untuk setiap mekanisme, identifikasi class-class, tampilan muka dan elemen yang lain yang berpartisipasi pada kolaborasi ini. Selanjutnya perlu diidentifikasi relasi diantara semua elemen tersebut.
- 3) Pertimbangkan satu skenario yang meliputi semua mekanisme tersebut. Pertahankan skenario tersebut sementara waktu dan ulangi untuk setiap obyek yang berpartisipasi pada mekanisme tersebut.
- 4) Ekspose nilai state dan attribute dari setiap obyek untuk memahami skenario tersebut.
- 5) Dengan cara yang sama, ekspose link diantara obyek-obyek yang mewakili instance dari asosiasi antara mereka.

### C. SOAL LATIHAN/TUGAS

1. Carilah definisi pengertian Use Case Diagram, Class Diagram, dan Object Diagram selain yang disebutkan diatas !
2. Definisikan pengertian Use Case Diagram, Class Diagram, dan Object Diagram menurut pendapatmu !
3. Buatlah studi kasus contoh untuk pembuatan Use Case Diagram !
4. Buatlah studi kasus contoh untuk pembuatan Class Diagram !
5. Buatlah studi kasus contoh untuk pembuatan Object Diagram !

### D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc.

John W. Satzinger, Robert B. Jackson, Stephen D. Burd (2016). *Systems Analysis and Design In A Changing World*. 7th edition. Boston: Cengage Learning.

## GLOSARIUM

**User** adalah istilah lain dari pengguna aplikasi.

**Use Case Diagram** adalah sebuah gambaran diagram *UML* yang dipakai untuk membuat perancangan sebuah sistem.

**Association** adalah sebuah fungsi perelasian untuk merelasikan hubungan antara sebuah element dengan element lainnya.

**Include** adalah sebuah fungsi perelasian yang digunakan ke sebuah use case untuk menjalankan sebuah fungsi.

**Extend** adalah sebuah fungsi perelasian yang digunakan ke sebuah use case yang nantinya use case tersebut dapat berfungsi tanpa bantuan use case tambahan.

**Attribute** adalah merupakan elemen atribut pada suatu kelas

**Object** adalah sebuah objek atau benda/element yang berada pada suatu diagram.

**Int** merupakan sebuah elemen yang memiliki type data angka.

**Char** merupakan sebuah elemen yang memiliki type data huruf.

**Operation** adalah suatu perintah yang dapat dilakukan pada sebuah kelas.

**Forward Engineering** merupakan perancangan dan implementasi hasil dari reverse engineering untuk menghasilkan sistem yang baru.

**Interface** adalah tampilan muka atau tampilan yang dapat dilihat oleh pengguna.