

## **BAB II**

### **LANDASAN TEORI**

#### **2.1. Pengertian Sistem Informasi**

Menurut Tri Utami dan Bambang Eka Purnama (2015) sistem informasi merupakan serangkaian komponen berupa manusia, prosedur, data, dan teknologi (seperti computer) yang digunakan untuk menghasilkan informasi yang bernilai untuk pengambilan keputusan. Sistem informasi yang melekat dan merupakan infrastruktur penunjang keberhasilan bagi setiap organisasi dalam mencapai tujuannya (*Bonnie Soeherman & Marion Pinontoan, 2004*)

#### **2.2. Pengertian Penjualan**

Menurut Angelina Permatasari (2013) Penjualan merupakan salah satu bisnis yang sangat penting bagi perusahaan. Penjualan adalah jumlah yang dibebankan kepada pelanggan untuk barang dagangan yang dijual, baik secara tunai maupun kredit (Warren, et al.,2005). Melalui penjualan, perusahaan akan menghasilkan profit yang optimal dalam jangka panjang. Oleh karena itu efektivitas dan efisiensi dari sistem informasi penjualan merupakan factor yang sangat penting bagi perusahaan dalam mencapai visi dan misi dari perusahaan itu sendiri.

#### **2.3. Pengertian Apotek**

Menurut Keputusan Menkes RI No.1332/Menkes/SK/X/2002 Apotek merupakan suatu tempat tertentu untuk melakukan pekerjaan kefarmasian dan penyaluran obat kepada masyarakat. Definisi apotek menurut PP 51 Tahun 2009 Apotek merupakan suatu tempat atau terminal distribusi obat perbekalan farmasi yang dikelola oleh apoteker sesuai standar dan etika kefarmasian.

## **2.4. Sejarah Singkat Apotek Kimia Farma**

Kimia Farma adalah perusahaan industri farmasi pertama di Indonesia yang didirikan oleh Pemerintah Hindia Belanda tahun 1817. Nama perusahaan ini pada awalnya adalah NV Chemicalien Handle Rathkamp & Co. Berdasarkan kebijaksanaan nasionalisasi atas eks perusahaan Belanda di masa awal kemerdekaan, pada tahun 1958, Pemerintah Republik Indonesia melakukan peleburan sejumlah perusahaan farmasi menjadi PNF (Perusahaan Negara Farmasi) Bhinneka Kimia Farma. Kemudian pada tanggal 16 Agustus 1971, bentuk badan hukum PNF diubah menjadi Perseroan Terbatas, sehingga nama perusahaan berubah menjadi PT Kimia Farma (Persero).

Pada tanggal 4 Juli 2001, PT Kimia Farma (Persero) kembali mengubah statusnya menjadi perusahaan publik, PT Kimia Farma (Persero) Tbk, dalam penulisan berikutnya disebut Perseroan. Bersamaan dengan perubahan tersebut, Perseroan telah dicatatkan pada Bursa Efek Jakarta dan Bursa Efek Surabaya (sekarang kedua bursa telah merger dan kini bernama Bursa Efek Indonesia). Berbekal pengalaman selama puluhan tahun, Perseroan telah berkembang menjadi perusahaan dengan pelayanan kesehatan terintegrasi di Indonesia. Perseroan kian diperhitungkan kiprahnya dalam pengembangan dan pembangunan bangsa, khususnya pembangunan kesehatan masyarakat Indonesia.

## **2.5. Visi & Misi Apotek Kimia Farma**

### **2.5.1. Visi**

Menjadi perusahaan Healthcare pilihan utama yang terintegrasi dan menghasilkan nilai yang berkesinambungan.

### **2.5.2. Misi**

1. Melakukan aktivitas usaha di bidang-bidang industri kimia dan farmasi, perdagangan dan jaringan distribusi, retail farmasi dan layanan kesehatan serta optimalisasi asset.

2. Mengelola perusahaan secara Good Corporate Governance dan operational excellence didukung oleh SDM professional.
3. Memberikan nilai tambah dan manfaat bagi seluruh stakeholder

## **2.6. Metode Pengembangan Sistem**

### **2.6.1. Pengertian Object Oriented Analysis and Design (OOAD)**

Object-Oriented Analysis and Design(OOAD) adalah metode untuk menganalisa dan merancang sistem dengan pendekatan berorientasi object.

Object diartikan sebagai suatu entitas yang memiliki identitas, state, dan behavior. Pada analisa, identitas sebuah object menjelaskan bagaimana seorang user membedakannya dari object lain, dan behavior object digambarkan melalui event yang dilakukannya. Sedangkan pada perancangan, identitas sebuah object digambarkan dengan cara bagaimana object lain mengenalinya sehingga dapat diakses,dan behavior object digambarkan dengan operation yang dapat dilakukan object tersebut yang dapat mempengaruhi object lain dalam sistem (Mathiassen (2000) dalam bukunya Object Oriented Analysis and Design).

Menurut Pungki Prayughi, Irman Hermandi, dan Heru Sukoco (2016), beberapa aktivitas utama yang dilakukan di dalam OOAD yakni (Satzinger 2007):

#### **1. Object, Attributes, and Methods**

Sebuah objek dalam sistem informasi adalah seperti sebuah objek di dunia nyata yaitu sesuatu yang memiliki *attributes* dan *behaviours*. Sebuah sistem informasi dapat memiliki berbagai jenis objek, seperti *User Interface* (UI) objek yang membentuk antarmuka pengguna dan sistem dan masalah objek domain yang menjadi fokus dari tugas lingkungan pengguna. Sebuah *User Interface* (UI) memiliki *Attributes*, yang merupakan karakteristik yang memiliki nilai : ukuran, bentuk, warna, lokasi, dan keterangan dari tombol atau tabel sebagai contohnya. Sebuah form pada layar memiliki atribut seperti tinggi dan lebar, gaya

perbatasan, dan warna latar belakang. Pengguna UI ini juga memiliki perilaku, atau metode yang menggambarkan apa yang objek dapat lakukan.

UI OBJECTS	ATTRIBUTES	METHODS
Button	size, shape, color, location, caption	click, enable, disable, hide, show
Label	size, shape, color, location, text	see text, get text, hide, show
Form	width, height, border style, background color	change size, minimize, maximize, appear, disappear

Gambar 2.1. Attributes and Methods of UI Objects.

Objek dari User Interface (UI) adalah yang paling mudah untuk dipahami karena pengguna dapat melihat mereka dan berinteraksi dengan mereka secara langsung. Tetapi sistem Object Oriented memuat jenis objek lainnya, yang disebut domain objek masalah, yang khusus dibuat untuk aplikasi bisnis.

## 2. Classes

Semua objek dari pelanggan diklasifikasikan sebagai jenis hal pelanggan, sehingga dalam pengembangan Object Oriented, dapat merujuk kepada kelas pelanggan ketika pengguna membicarakan tentang semua objek pelanggan. Kelas mendefinisikan apa semua objek dari kelas mewakili. Ketika pengguna bicara tentang pemrograman komputer dan benda-benda, anda dapat mengacu kepada objek sebagai contoh kelas.

## 3. Inheritance and polymorphism

Mungkin sebuah konsep yang paling sering digunakan adalah ketika membahas objek kelas adalah objek pewarisan. Dimana suatu objek kelas

mengambil karakteristik kelas lain. Sebagai contoh, sebuah objek memiliki kelas nasabah mungkin juga sesuatu yang lebih umum, seperti orang. Oleh karena itu, jika kelas orang sudah didefinisikan, kelas pelanggan dapat didefinisikan dengan memperluas kelas pelanggan untuk mengambil atribut yang lebih spesifik dan metode lainnya yang diperlukan pelanggan.

#### 2.6.2. Prinsip umum OOAD

1. Model the context : Sistem yang bermanfaat sesuai dengan konteks OOAD.
2. Emphasize the architecture : Merupakan arsitektur yang mudah dipahami yang memfasilitasi kolaborasi antara designer dan programmer.
3. Reuse Patterns : Dibangun berdasarkan gagasan – gagasan yang kuat dan komponen pretests memperbaiki kualitas sistem dan produktivitas dari proses development.
4. Tailor the method specific projects : Setiap usaha development masing masing mempunyai tantangan yang unik. OOAD harus disesuaikan dengan kebutuhan – kebutuhan yang khusus dari situasi analisis dan design yang diberikan.

#### 2.6.3. Teknik pemodelan yang ada pada OOAD

1. Model Objek :
  - a. Model objek menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya.
  - b. Model objek berisi diagram objek. Diagram objek adalah graph dimana nodenya adalah kelas yang mempunyai relasi antar kelas.
2. Model Dinamik
  - a. Model dinamik menggambarkan aspek dari sistem yang berubah setiap saat.

- b. Model dinamik dipergunakan untuk menyatakan aspek control dari sistem.
  - c. Model dinamik berisi state diagram. State diagram adalah graph dimana nodenya adalah state dan arc adalah transisi antara state yang disebabkan oleh event.
3. Model Fungsional
- a. Model fungsional menggambarkan transformasi nilai data di dalam sistem.
  - b. Model fungsional berisi data flow diagram. DFD adalah suatu graph dimana nodenya menyatakan proses dan arcnya adalah aliran data.

#### 2.6.4. Kelebihan dan kelemahan OOAD

##### 1. Kelebihan

Dibandingkan dengan metode SSAD, OOAD lebih mudah digunakan dalam pembangunan sistem dibandingkan dengan SSAD, waktu pengembangan, level organisasi, ketangguhan, dan penggunaan kembali (reuse) kode program lebih tinggi dibandingkan dengan metode OOAD. Tidak ada pemisahan antara fase desain dan analisis, sehingga meningkatkan komunikasi antara user dan delveloper dari awal hingga akhir pembangunan sistem.

##### 2. Kelemahan

Pada awal desain OOAD sistem mungkin akan sangat simple. Pada OOAD lebih fokus pada koding dibandingkan dengan SSAD. Pada OOAD tidak menekankan pada kinerja team seperti pada SSAD. Pada OOAD tidak mudah untuk mendefinisikan class dan objek yang dibutuhkan sistem. Sering kali pemrograman berorientasi objek digunakan untuk melakukan analisis terhadap fungsional site sementara metode OOAD tidak berbasis pada fungsional sistem.

### 2.6.5. Konsep dasar OOAD

Beberapa konsep OOAD adalah sebagai berikut :

#### 1. Objek (Object)

Objek adalah benda secara fisik dan konseptual yang ada di sekitar kita. Sebuah objek memiliki keadaan sesaat yang disebut state.

#### 2. Kelas (Class)

Kelas merupakan gambaran sekumpulan objek yang terbagi dalam atribut, operasi, metode, hubungan, dan makna yang sama.

#### 3. Kotak hitam (Black Boxes)

Sebuah objek adalah kotak hitam. Konsep ini menjadi dasar implementasi objek. Dalam operasi OO hanya developer yang dapat memahami detail proses yang ada didalam kotak tersebut, sedangkan user tidak perlu mengetahui apa yang dilakukan yang penting mereka dapat menggunakan objek untuk memproses kebutuhan mereka. Kotak hitam berisi kode dan data.

#### 4. Asosiasi dan agregasi

Asosiasi adalah hubungan yang mempunyai makna antara sejumlah objek. Asosiasi digambarkan dengan sebuah garis penghubung diantara objeknya. Agregasi adalah bentuk khusus sebuah asosiasi yang menggambarkan seluruh bagian pada suatu objek merupakan bagian dari objek yang lain.

### 2.6.6. Tahapan-tahapan OOAD

#### 1. Menentukan kebutuhan pemakai

Mengidentifikasi proses proses bisnis.diperlukan karena dapat menjelaskan aktivitas-aktivitas apa saja yang akan dikerjakan oleh sistem dan juga perilaku dari komponen-komponan sistem.

2. Identifikasi skenario pemakai

Skenario pemakai digunakan untuk merepresentasikan sebuah interaksi antara aktor dan sistem.

3. Identifikasi kelas dan objek.

Mengidentifikasi kelas dan objek yang ada dalam lingkup sistem. Kelas dan objek dapat diidentifikasi dari entitas eksternal yang memproduksi dan memakai informasi yang akan digunakan oleh sistem, sesuatu yang merupakan bagian dari wilayah informasi permasalahan, serta kejadian misalnya prosedur operasional yang muncul dalam lingkup operasional sistem.

4. Identifikasi atribut dan operasi kelas.

Identifikasi atribut dan operasi kelas dapat mengacu pada skenario kelas yang telah ditentukan. Atribut diidentifikasi dari elemen-elemen data yang menggambarkan sebuah objek secara utuh.

5. Mendefinisikan struktur dan hirarki kelas.

Identifikasi struktur dan hirarki kelas dapat dihasilkan berdasarkan identifikasi kelas dan operasi pada kelas. Mendefinisikan struktur dan hirarki kelas dapat mengatur dan menyederhanakan objek-objek menjadi kelas-kelas objek.

6. Membangun model keterhubungan kelas dan objek.

Keterhubungan antara kelas dan objek berdasarkan pada identifikasi kelas, sehingga dapat menentukan hubungan antara kelas satu dengan kelas lainnya.

7. Perancangan berbasis objek.

Perancangan berbasis objek biasa dilakukan setelah tahapan analisis



## 2.7. Unified Modelling Language (UML)

*UML (Unified Modelling Language)* merupakan satu kumpulan konvensi pemodelan yang digunakan untuk merincikan atau menjelaskan sistem piranti lunak yang dijadikan sebagai standar pemodelan objek. *UML* dapat dijadikan panduan bagi *developer* dalam mengetahui sudut pandang sebuah sistem dan dikomunikasikan kepada pihak – pihak yang terkait dengan sistem dalam bentuk diagram. *UML* merupakan bahasa kesatuan yang mendeskripsikan model sebuah sistem secara efektif.

### 2.7.1. Use Case Diagram

*Use case diagram* merupakan diagram yang menggambarkan interaksi antara sistem, pengguna atau dengan sistem eksternal lainnya. Dengan kata lain, menggambarkan siapa pengguna sistem dan bagaimana pengguna berinteraksi dengan sistem.

Notasi dan relasi yang digunakan dalam menggambarkan *use case diagram* yaitu:

#### 1. *Use Case*

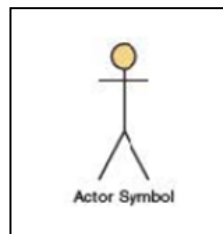
*Use case* merupakan suatu tugas bisnis yang diselesaikan dengan urutan langkah-langkah perilaku baik secara otomatis atau manual. *Use case* mendeskripsikan fungsi - fungsi yang ada pada sistem dari sudut pandang para pengguna eksternal, dalam cara dan terminologi yang mereka mengerti.



Gambar 2.2. Notasi *Use Case* pada *Use Case Diagram*

## 2. *Aktor*

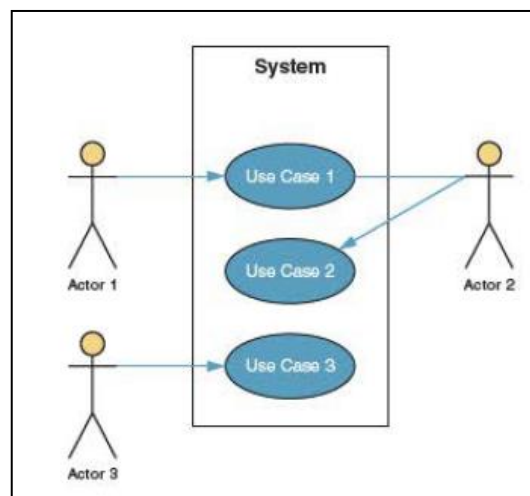
Digambarkan dalam bentuk manusia, aktor merupakan segala sesuatu yang berinteraksi dengan sistem untuk bertukar informasi.



Gambar 2.3. Notasi Aktor pada *Use Case Diagram*

## 3. *Relationship*

Digambarkan dengan sebuah garis yang menunjuk atau menghubungkan antara sebuah *use case* dengan aktor atau dengan *use case* lainnya, yang mendefinisikan hubungan interaksi di antaranya.



Gambar 2.4. Interaksi Aktor dengan *Use Case*

#### 4. *Association*

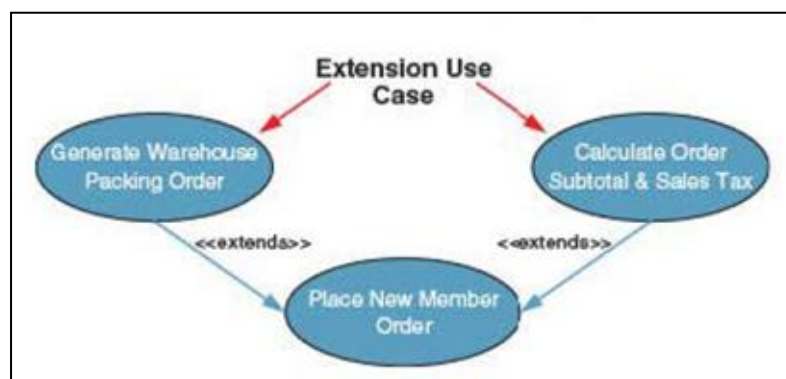
Relasi antara aktor dan *use case* dimana interaksi terjadi antara keduanya. Digambarkan dengan sebuah garis dengan panah yang menyentuh *use case* atau garis biasa yang menghubungkan antara notasi aktor dengan notasi *use case*. Garis panah dari aktor menuju *use case* berarti aktor memberikan data ke fungsi dalam *use case*, sedangkan garis biasa berarti aktor hanya menerima atau membaca data dari *use case*.



Gambar 2.5. Contoh Relasi *Association*

#### 5. *Extends*

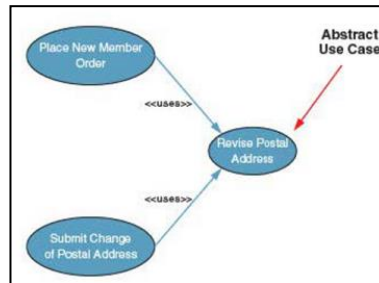
Relasi *extends* memperluas fungsionalitas *use case* apabila kondisi tertentu dipenuhi.



Gambar 2.6. Contoh Relasi *Extends*

## 6. *Uses* atau *Includes*

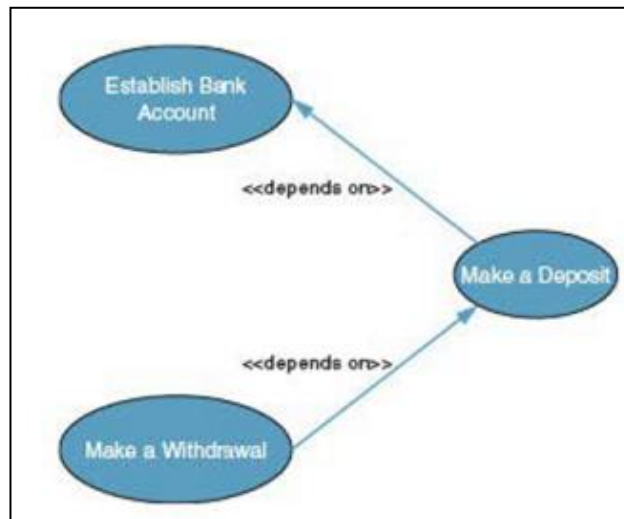
Relasi *uses* mengurangi pengulangan antara dua atau lebih *use case* dengan menggabungkan langkah-langkah dalam *use case*.



Gambar 2.7. Contoh Relasi *Uses* atau *Include*

## 7. *Depends On*

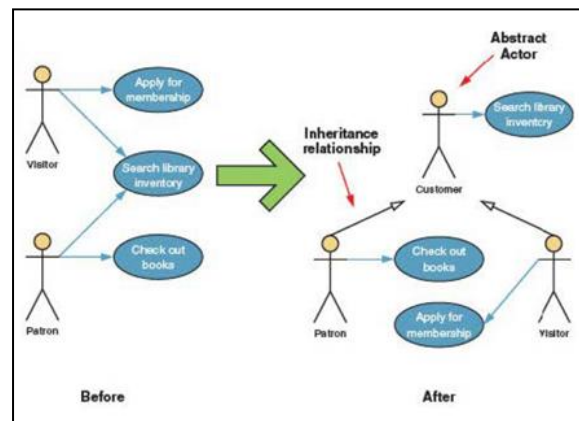
Relasi antara *use case* yang mengindikasikan bahwa *use case* tidak dapat dilakukan sampai *use case* lainnya sudah dijalankan.



Gambar 2.8. Contoh Relasi *Depends On*

## 8. *Inheritance*

Relasi *inheritance* digunakan saat dua atau lebih aktor menggunakan *use case* yang sama.



Gambar 2.9. Contoh Relasi *Inheritance*

### 2.7.2. Activity Diagram

*Activity diagram* digambarkan untuk memperlihatkan proses alur bisnis, langkah - langkah *use case*, atau logika perilaku objek. *Activity diagram* serupa dengan *flow chart diagram* dalam menggambarkan alur aktivitas yang berurutan dari proses bisnis atau *use case*. Namun, berbeda dari *flow chart* karena memberikan mekanisme yang menggambarkan aktivitas yang terjadi secara paralel. Oleh karena itu, *activity diagram* berguna untuk memodelkan aksi-aksi saat operasi dieksekusi dan memodelkan hasil dari aksi tersebut.

Notasi yang digunakan dalam *activity diagram* yang biasa dipakai adalah sebagai berikut:

#### 1. *Initial Node*

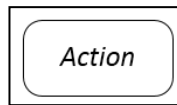
Digambarkan dengan simbol lingkaran penuh, merepresentasikan dimulainya sebuah proses.



Gambar 2.10. Simbol *Initial Node* pada *Activity Diagram*

## 2. *Action*

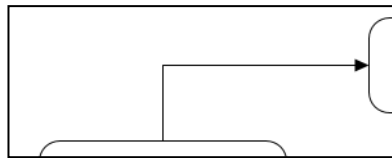
Digambarkan dengan simbol persegi panjang melengkung (*rounded rectangle*) yang merepresentasikan langkah individual. Urutan dari *actions* menunjukkan kesatuan *activity* pada diagram.



Gambar 2.11. Simbol *Action* pada *Activity Diagram*

## 3. *Flow*

Digambarkan dengan panah yang mengindikasikan adanya proses yang terjadi sepanjang *action*.



Gambar 2.12. Simbol *Flow* pada *Activity Diagram*

## 4. *Decisions*

Digambarkan dalam bentuk belah ketupat dengan 1 panah masuk dan 2 atau lebih panah keluar, mengindikasikan sebuah kondisi yang terjadi. Bentuk *node* ini dapat dilihat pada gambar 2.13.

5. *Merge*

Digambarkan dalam bentuk belah ketupat dengan 2 atau lebih panah masuk yang sebelumnya dipisahkan oleh *decision*.



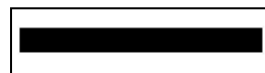
Gambar 2.13. Simbol *Decision / Merge* pada *Activity Diagram*

6. *Fork*

Digambarkan dalam sebuah persegi panjang hitam dengan 1 alur masuk dan 2 atau lebih alur keluar. *Fork* merepresentasikan *action* yang terjadi secara bersamaan atau paralel. Bentuk *node* ini dapat dilihat pada gambar 2.14.

7. *Join*

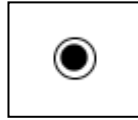
Digambarkan dalam sebuah persegi panjang hitam dengan 2 atau lebih alur masuk yang sebelumnya terjadi secara paralel dan 1 alur keluar untuk melanjutkan proses pada *activity diagram*.



Gambar 2.14. Simbol *Fork / Join* pada *Activity Diagram*

8. *Activity final*

Digambarkan dengan sebuah lingkaran penuh di dalam sebuah lingkaran berongga, merepresentasikan berakhirnya sebuah proses sebuah *activity diagram*.



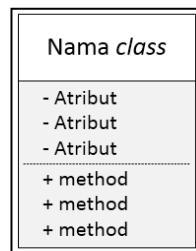
Gambar 2.15. Simbol *Final Node* pada *Activity Diagram*

### 2.7.3. Class Diagram

Sebuah *class diagram* digunakan untuk menggambarkan objek dan hubungan antar objek. Elemen – elemen yang ada di dalam *class diagram*, yaitu:

#### 1. *Class*

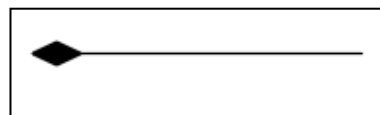
Sebuah *class* digambarkan sebagai sebuah kotak yang terbagi atas tiga bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan atribut *class*. Bagian akhir mendefinisikan perilaku dari sebuah *class*.



Gambar 2.16. Elemen *Class* pada *Class Diagram*

#### 2. *Association*

Dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempat dia bergantung tersebut. Sebuah relasi *composition* digambarkan sebagai garis dengan ujung berbentuk belah ketupat penuh.

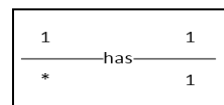


Gambar 2.17 Elemen *Composition* pada *Class Diagram*



### 3. *Dependency*

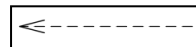
Pada saat tertentu sebuah *class* menggunakan *class* yang Asosiasi merupakan sebuah *relationship* paling umum antara dua *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara dua *class*. Garis ini bisa melambangkan tipe - tipe *relationship* dan juga dapat menampilkan hukum-hukum *multiplicity* pada sebuah *relationship*, seperti *one-to-one*, *one-to-many*, dan *many-to-many*.



Gambar 2.18 Elemen *Association* pada *Class Diagram*

### 4. *Composition*

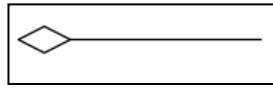
Jika sebuah *class* tidak bisa berdiri sendiri lain, hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah dengan garis putus - putus.



Gambar 2.19 Elemen *Dependency* pada *Class Diagram*

### 5. *Aggregation*

Agregasi merupakan tipe relasi yang unik, yang salah satu objeknya adalah “bagian” dari objek lainnya.

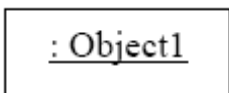

Gambar 2.20 Elemen *Aggregation* pada *Class Diagram*



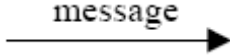
#### 2.7.4. Sequence Diagram

*Sequence diagram* menggambarkan interaksi antar-objek melalui pesan di dalam eksekusi sebuah *use case* atau operasi. Sistem *sequence diagram* membantu mengidentifikasi pesan *high level* yang masuk dan keluar dari sistem. Kemudian, pesan ini akan digunakan masing-masing objek untuk berkomunikasi dengan objek-objek lainnya.

Notasi yang digunakan dalam *sequence diagram* adalah *actor*, *system*, *lifelines*, *activation bars*, *input messages*, dan *output messages*.

Tabel 2.1. Notasi/symbol Sequence Diagram

SIMBOL	NAMA	KETERANGAN
	<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.

## 2.8. PHP

Menurut Yuntari Purba Sari (2017), “PHP adalah singkatan dari PHP *Hypertext Preprocessor*. Saat pertama kali dikembangkan oleh programmer bernama Rasmus Lerdorf, PHP awalnya adalah singkatan dari *Personal Home Page Tools*” (Tim EMS, (2014:59)).

## 2.9. MySQL

Menurut Yuntari Purba Sari (2017), “MySQL adalah nama database server. Database server adalah server yang berfungsi untuk menangani database” (Kadir, (2013:15)).