# MODUL

# PERANCANGAN SISTEM BERORIENTASI OBJECK

**Disusun oleh**

**Ani Yoraeni, S.Pd, M.Kom**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER NUSA MANDIRI**
**JAKARTA**

**2019**

# KATA PENGANTAR

Puji syukur alhamdulillah saya panjatkan ke hadirat Tuhan Yang Maha Esa, karena telah melimpahkan rahmat-Nya berupa kesempatan dan pengetahuan sehingga Modul ini bisa selesai pada waktunya.

Saya berharap semoga Modul ini bisa menambah pengetahuan para pembaca. Meskipun saya sangat berharap agar Modul ini tidak memiliki kekurangan, tetapi saya menyadari bahwa pengetahuan saya sangatlah terbatas. Oleh karena itu, saya tetap mengharapkan masukan serta kritik dan saran yang membangun dari pembaca untuk Modul ini demi terlaksananya pembuatan Modul menganalisa dengan baik, sehingga tujuan untuk proses pembuatan Modul ini juga bisa tercapai.

Semoga adanya Modul ini bisa memberikan wawasan lebih luas lagi dan juga menjadi sebuah sumbangan pemikiran kepada para pembaca dan khususnya untuk para mahasiswa Nusa Mandiri, Aamiin. Saya menyadari bahwa isi atau kata dari Modul ini masih banyak kekurangan-kekurangan dan jauh dari kata sempurna. Maka dari itu, kepada bapak/Ibu dosen pembimbing, saya meminta sedikit kritikan dan sarannya guna untuk memperbaiki pembuatan Modul di masa yang akan datang.

## DAFTAR ISI

Halaman

# BAB I
# OBJECT ORIENTED

## 1. Sejarah Object Oriented

Sejarah awal object oriented berawal dari Konsep awal programming (Basic) yang dimulai dari kekuatan GOTO statement, yang merupakan Non Procedural Language. Procedural Language/Bahasa pemograman terstruktur yaitu dengan menghilangkan kelemahan GOTO konsep non procedural language .Contohnya seperti : Pascal, COBOL, FORTRAN, BASIC dan lain lain.Object Oriented Programming ini mengarah pada konsep object. Pada Akhir tahun 1960 diperkenalkan pertama kali dengan bahasa SIMULA. Kemudian pada tahun 1970 dikembangkan Smaltalk yaitu Bahasa pemrograman lainnya seperti Clipper 5.2 Java, Prolog dan lain lain .
Kemudian pada tahun 1991 Visual Object Oriented Programming, diperkenalkan pertama kali dengan bahasa Visual Basic oleh Microsoft . Bahasa pemograman lainnya : Visual C++, Visual Foxpro 3.0, CORBA (Common Object Request Broker Architecture), dan lain lain.

Pengembangan berorientasi objek merupakan cara berpikir baru tentang perangkat lunak yang berdasarkan pada abstraksi yang terdapat dalam dunia nyata. Dalam konteks pengembangan menunjukan pada bagian awal dari siklus hidup pengembangan sistem, yaitu dimulai dari survei, analisa, desain, implementasi dan pemeliharaan sistem. Konsep mengidentifikasi dan mengorganisasi domain aplikasi dari pada penggunaan bahasa pemrograman, berorientas object atau tidak merupakan hal yang sangat penting dalam pengembangan berorientasi objek.

## 2. Konsep Pemodelan

Pada Pemodelan Berorientasi Objek terdiri dari pengembangan representasi objek secara progresif melalui tiga fase yaitu analisis, desain, dan implementasi. Pada tahap awal pengembangan, fokus utama adalah rincian eksternal sistem dan model yang dikembangkan adalah abstrak. Model tersebut akan menjadi lebih rinci ketika mengalami revolusi, sedangkan fokus sentralnya bergeser ke arah pemahaman bagaimana sistem itu akan dibangun dan berfungsi dengan baik. Teknik pemodelan itu kemudian akan diimplementasikan dengan menggunakan bahasa yang mendukung model pemrograman berorientasi objek. Dimana model objek harus menggambarkan objek dalam sistem dan relasinya, model dinamik yang

menggambarkan interaksi antara objek pada sistem, sedangkan model fungsional menggambarkan transformasi data dalam sistem.

Teknik pemodeIan objek tebagi menjadi tiga macam modeI untuk menggambarkan sistem, yaitu:

1. **Model Objek** yaitu suatu model yang menggambarkan struktur statis dari objek dalam sistem dan reIasinya. yang berisikan diagram objek.
2. **Model fungsional** yaitu suatu model yang menggambarkan transformasi niIai data di daIam sistem yang berisi data fIow diagram (diagram aIir data.
3. **ModeI dinamik** yaitu suatu model yang menggambarkan aspek dari sistem yang berubah setiap saat dan digunakan untuk menyatakan aspek kontroI dari sistem yang berisi state diagram..

Konstruksi objek menggunakan kumpulan objek yang berisi niIai-niIai tersimpan dari variabeI instan yang ditemukan daIam suatu objek. Tidak seperti modeI yang berorientasi pada rekaman, nilai-nilai berorientasi objek adaIah objek semata-mata. Pendekatan pemodelan berorientasi objek menciptakan penyatuan aplikasi dan pengembangan basis data dan mengubahnya menjadi model data terpadu dan lingkungan bahasa. Pemodelan berorientasi objek memungkinkan identifikasi objek dan komunikasi sambil mendukung abstraksi data, pewarisan dan enkapsulasi**.**

## 3. Metodelogi pemodelan objeck

Metodologi berorientasi objek diperkenalkan pada tahun 1980, menggunakan perangkat kerja dan teknik-teknik yang dibutuhkan dalam pengembangan sistem, yaitu dynamic dan static object oriented model, state transition diagram dan case scenario. Fokus utama metodologi ini pada objek, dengan melihat suatu sistem terdiri dari objek yang saling berhubungan. Objek dapat digambarkan sebagai benda, orang, tempat dan sebagainya yang mempunyai atribut dan metode. Metodologi terdiri dari pembuatan model dari domain aplikasi, kemudian menambahkan rincian implementasi pada saat pembuatan desain dari suatu sistem. Tahap-tahap metodologi berdasarkan System Development Life Cycle(SDLC) digunakan dengan memperhatikan karakteristik khusus berorientasi objek, sbb:

a. Project Management & Planning
b. Analisis
c. Desain

d. Implementasi dan operasi

## 4. Model Object Oriented

### 4.1 Objek dan kelas

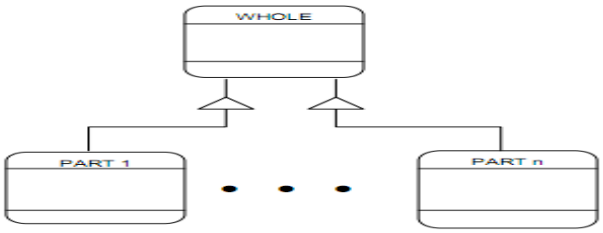
Objek adalah entitas runtime dasar yang dapat diidentifikasi dalam OOP. Objek dapat mewakili entitas dunia nyata, seperti orang, mobil, tempat, dan lain-lain. Misalnya, kita dapat mengatakan mobil adalah objek yang memiliki beberapa karakteristik khusus, seperti jumlah gear, warna serta memiliki beberapa fungsi seperti pengereman, akselerasi, dan sebagainya. Jadi, suatu objek mewakili suatu entitas yang dapat menyimpan data dan memiliki interface melalui fungsi.

Kelas adalah struktur pembuatan tipe data yang ditentukan pengguna yang menampung seluruh kumpulan data suatu objek. Jadi, kelas pada dasarnya adalah template dari serangkaian objek yang berbagi beberapa *properties* dan *behaviour*. Kita juga bisa mengatakan bahwa kelas adalah kumpulan objek bertipe serupa. Setelah kelas dibuat, kita dapat membuat sejumlah objek milik kelas itu. Sebenarnya, kelas tidak menentukan data apapun. Kelas menentukan properti atau metode yang mungkin berisi dalam objek dari kelas itu. Kelas biasanya bertindak seperti tipe data bawaan tetapi sebenarnya tipe data yang ditentukan pengguna.

## 5. Struktur Objek dan Hirarki Kelas

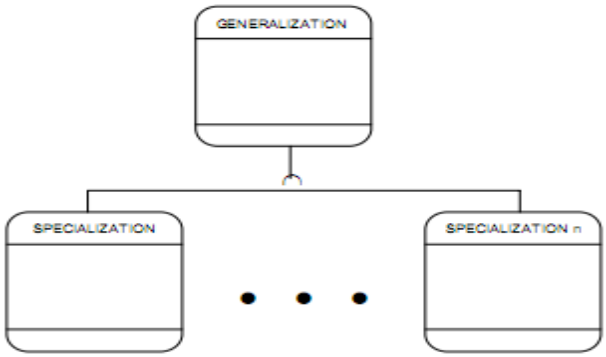
### 5.1 Whole-Part Structure

Memperlihatkan hirarki dari suatu kelas sebagai komponen dari kelas lain yang disebut juga sub objek. Contohnya, kelas Mobil adalah Whole dan komponennya Mesin, Rangka, dll merupakan Part1, Part2,…,Part n..



### 5.2. Gen-Spec

Memperlihatkan kelas sebagai spesialisasi dari kelas diatasnya. Kelas yang mempunyai sifat umum disebut Generalization, Superclass atau Topclass. Kelas yang mempunyai sifat khusus disebut Specialization.Contohnya kelas Mobil adalah Generalization,Sedan, Truk, Minibus, dll merupakan Specizlization 1,Specialization 2, dst.



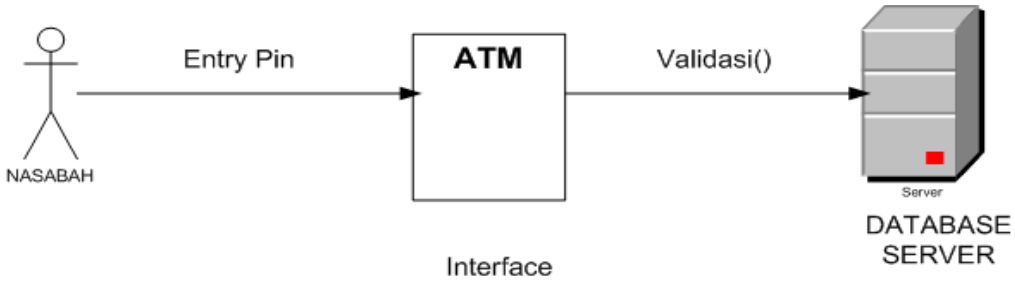

### 6. Karakteristik Berorientasi Objek

**B**erorientasi objek mempunyai karakteristik sebagai berikut:

**a. Encapsulation (pembungkusan)**

- Pengkapsulan merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Dengan demikian objek atau prosedur dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.
- Sebuah object yang terkapsulasi dapat dianggap sebagai *black box.*
- Proses di dalamnya adalah tertutup bagi klien, yang hanya memanggil metode yang menjadi interface.
- Dalam Java, dasar enkapsulasi adalah Class. Variabel atau method sebuah class tidak dapat diakses dengan menjadikan class tersebut private/protected.

Contoh Enkapsulas




Disini terjadi penyembunyian informasi tentang bagaimana cara kerja pengecekan validitas kartu, kecocokan pin yang dimasukkan, koneksi ke database server, dll, dimana hal-hal tersebut tidak perlu diketahui oleh pengguna tentang bagaimana cara kerjanya

**b. Inheritance (pewarisan)**

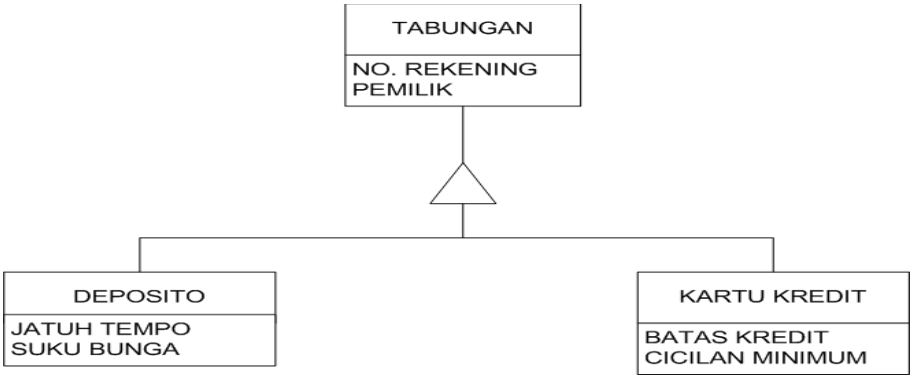
Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Bila inheritance dipergunakan, kita tidak perlu membuat atribut dan metode lagi pada anaknya, karena telah diwarisi oleh induknya. Inheritance mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di antara class yang mempunyai hubungan secara hirarki.

- Sebuah class bisa mewariskan atribut dan method-nya ke class yang lain

–Class yang mewarisi disebut superclass

–Class yang diberi warisan disebut subclass

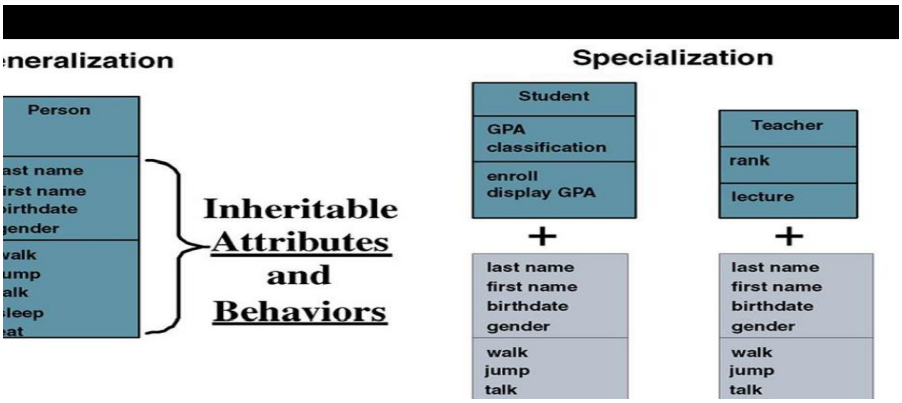
–Sebuah subclass bisa mewariskan atau berlaku sebagai superclass bagi class yang lain => disebut multilevel inheritance




**c. Generalisasi**

- Generalisasi adalah relasi antara beberapa subclass dengan superclass di atasnya.

- Kelas yang lebih rendah mewarisi semua atribut yang dimiliki oleh kelas yang lebih tinggi dan juga memiliki atribut yang membedakannya dengan kelas-kelas lain yang sederajat.



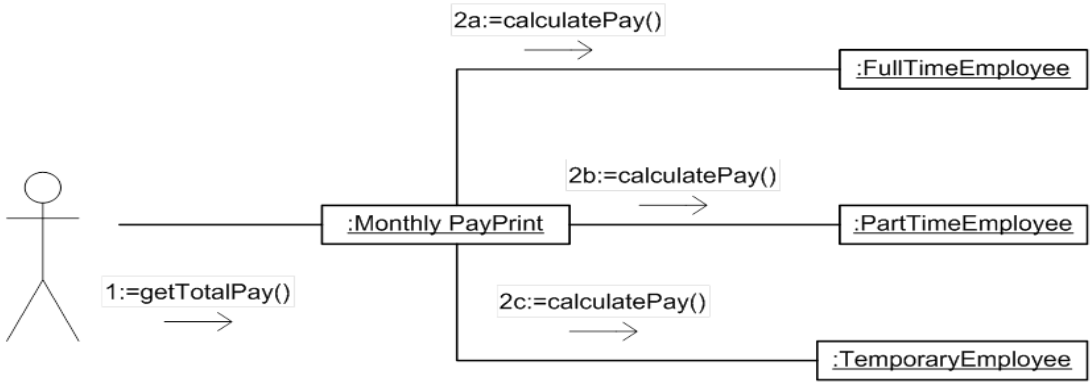

**3. Polymorphisme**

Polymorphisme yaitu aksi yang sama yang dapat dilakukan terhadap beberapa objek. Polimorfisme berarti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

•Polymorphism adalah kemampuan untuk tampil dalam berbagai bentuk.

•Hal ini mengacu pada kemungkinan message yang sama dikirimkan ke obyek-obyek lain pada class yang berbeda, dan merespon secara berbeda.

Contoh Polymorphism



**7. Perbedaan Object Oriented dengan Non Object**

a. Penggunaan alat

Metodologi non objek menggunakan beberapa alat untuk menggambarkan model seperti data flow diagram, entity relationship diagram dan structure chart. Sedangkan metodologi berorientasi objek menggunakan satu jenis model dari tahap analisa sampai implementasi, yaitu diagram objek.

b. Data dan proses

Pada metodologi non objek, data dan proses dianggap sebagai dua komponen yang berlainan Sedangkan pada metodologi berorientasi objek, data dan proses merupakan satu kesatuan, yaitu bagian dari objek.

c. Bahasa pemrograman

Metodologi non objek dipergunakan untuk melengkapi pemrograman terstruktur pada bahasa generasi ketiga. Sedangkan metodologi berorientasi objek dipergunakan untuk pemrograman berorientasi objek dan bahasa generasi keempat.

# BAB II

# UML (Unified Modeling Language)

## 1. Pengertian UML

UML atau "Unified Modelling Language" adalah suatu metode permodelan secara visual yang berfungsi sebagai sarana perancangan sistem berorientasi objek.

Definisi UML adalah sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan, dan juga pendokumentasian sistem aplikasi. Saat ini UML menjadi bahasa standar dalam penulisan blue print software (arsitektur).

Menurut (Rosa-Salahuddin, 2011:113), Unified Modelling Language atau UML merupakan salah satu standar bahasa yang banyak digunakan di dunia industri untuk menggambarkan kebutuhan (requirement), membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (PBO).

## 2. Sejarah UML

UML dimulai secara resmi pada Oktober 1994, ketika Rumbaugh menggabungkan kekuatan dengan Booch. Mereka berdua lalu bekerja bersama di Relational Software Cooperation. Proyek ini memfokuskan pada penyatuan metode booch dan Rumbaugh(OMT). Pada bulan October 1995, UML merilis versi 0.8 dan pada waktu yang sama juga Jacobson bergabung dengan Relational. Cakupan dari UML pun semakin meluas. Kemudian dibangunlah persatuan untuk UML dengan beberapa organisasi yang akan menyumbangkan sumber dayanya untuk bekerja, mengembangkan,dan melengkapi UML

Banyak partner yang berkontribusi pada UML 1.0, diantaranya Digital Equipment Corporation, Hawlett-Packard, I-Logix, IBM, ICON Computing, MCI systemhouse, Microsoft, Oracle, Relation, Texas Insturments dan Unisys. Dari kolaborasi ini dihasilkan UML 1.0 yang merupakan bahasa pemodelan yang ditetapkan secara baik, expressive, kuat dan cocok untuk lingkungan masalah yang luas. Dan pada January 1997, UML dijadikan sebagai standar bahasa pemodelan.

## 3. Konsep Dasar UML

**UML** merupakan sebuah standar untuk merancang sebuah model sistem, untuk mengembangkan **UML** maka harus di perhatikan diagram seperti apa yang di butuhkan dalam perancangan untuk membuat model sistem yaitu sebagai berikut :

1. Diagram Use Case
2. Diagram Aktifitas
3. Diagram Sekuensial
4. Diagram Kolaborasi
5. Diagram Kelas
6. Diagram Statechart
7. Diagram Komponen
8. Diagram Deployment
9. Diagram Object

Dari 9 diagram tersebut, dalam pemodelan sistem tidak harus di buat semua, minimal ada 4 diagram terpenting yang harus di buat yaitu sebagai berikut :

- Diagram Use Case
- Diagram Aktifitas
- Diagram Sequensial
- Diagram Kelas

***Use Case***

Diagram Use Case merupakan suatu pemodelan sistem yang terdiri dari aktor lalu di hubungkan dengan use case pada sistem yang di buat, diagram ini menggambarkan bagaimana keterhubungan antara aktor dengan use case, aktor disini tidak selalu berupa manusia sebagai pengelola ataupun customer, bisa juga berupa peralatan ataupun suatu sistem lain yang berhubungan dengan sistem yang kita kerjakan saat ini.

***Diagram Aktivitas***

Diagram Activity bisa juga di sebut flowmap ini merupakan pemodelan berupa arus flow yang di mulai dari tanda start sampai end ciri dari flowmap itu berupa suatu kumpulam entitas yang melakukan proses sistem sebagaimana mestinya yang saling terhubung.

***Diagram Sequensial***

Diagram ini menggambarkan suatu aliran fungsi dari use case, jadi intinya suatu aliran yang berfungsi sebagai pemodelan yang di hubungkan dengan use case dan fungsi nya masing masing, agar bisa di lihat fungsi satu use case dengan use case lainnya.

***Diagram Kelas***

Diagram kelas (Class Diagram) itu merupakan penggambaran dari sistem itu sendiri yang berupa program ataupun suatu menu form pilihan yang berisi tentang apa-apa saja yang saling berdekatan di gambarkan dengan class diagram berupa tabel kotak-kotak kecil berisi suatu kata kunci dan isi nya masing-masing yang berhubungan, dalam merancang class diagram ini harus membutuhkan ketelitian dan pemikiran yang dapat di implementasikan kedalam program sistem yang akan di buat nanti.

**BAB III**

# ACTIVITY DIAGRAM

1. **Pengertian Activity diagram**, sesuai dengan namanya diagram ini menggambarkan tentang aktifitas yang terjadi pada sistem. Dari pertama sampai akhir, diagram ini menunjukkan langkah – langkah dalam proses kerja sistem yang kita buat. Sebagai contoh, langkah – langkah memasak air. Tetapi kita akan menjelaskannya dengan bentuk grafik. Struktur diagram ini juga mirip dengan flowchart.

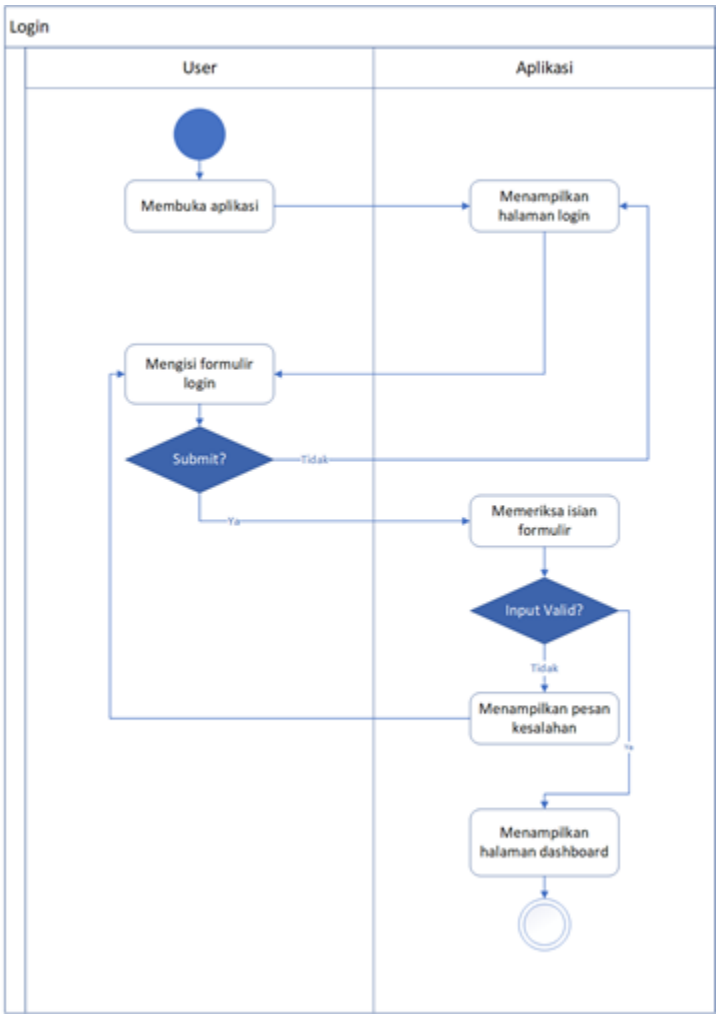
## 2. Fungsi Activity Diagram

- Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses
- Memperlihatkan urutan aktifitas proses pada sistem
- Activity diagram dibuat berdasarkan sebuah atau beberapa use case pada use case diagram

Berikut merupakan komponen penyusun activity diagram ini :

| Nama | Simbol | Fungsi |
|---|---|---|
| Initial State |  | Menggambarkan awal dimulainya suatu aliran aktivitas |
| Final State |  | Menggambarkan berakhirnya suatu aliran aktivitas |
| Activity |  | Menggambarkan aktivitas yang dilakukan dalam suatu aliran aktivitas |
| Decision |  | Menggambarkan pilihan kondisi atau cabang-cabang aktivitas tertentu |
| Transition |  | Berguna untuk menghubungkan satu komponen dengan komponen lainnya. |

**Contoh activity diagram :**



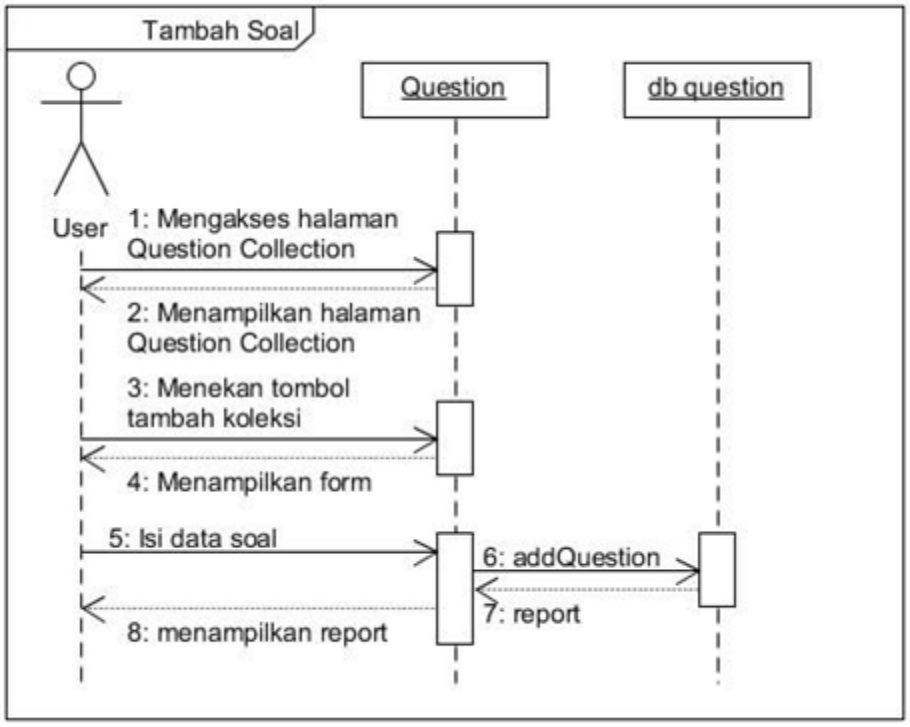

Gambar di atas merupakan Activity Diagram dari fungsi "Login". Seperti terlihat pada gambar, terdapat 2 *swimlane*, yaitu User dan Aplikasi. Tahap-tahapnya dibuat sangat rinci, aksi apa yang dilakukan user, akan menimbulkan reaksi dari aplikasi itu sendiri

### 3. Sequence Diagram

Menggambarkan sejumlah aktivitas atau kolaborasi antar *object*. Fungsi dari diagram ini yaitu untuk menunjukkan interaksi atau pesan yang disampaikan antar setiap *object*nya. Komponen-komponen penyusun Sequence Diagram ini yaitu:

| Nama | Simbol | Fungsi |
|---|---|---|
| Object |  | Menggambarkan sebuah *class* atau *object*. |
| Activation boxes |  | Menggambarkan panjang waktu yang dibutuhkan sebuah *object* dalam mengerjakan tugasnya |
| Actors |  | Menggambarkan pengguna yang berinterakasi dengan sistem |
| Lifeline |  | Menggambarkan "garis hidup" sebuah *object* |
| Message |  | Menggambarkan pesan atau interaksi antar *object* |
| Message to Self |  | Menggambarkan pesan balikan atau reaksi dari *object* sebelumnya |

Berikut contoh squance diagram :




Gambar di atas merupakan Sequence Diagram dari fitur Tambah Soal. Terdapat satu actor bernama "User" dengan dibersamai oleh *object* "Question" dan *object* "db_question". Setiap

tahapnya diwakili oleh tanda panah bergaris tegas (message) dan tanda panah bergaris putus-putus (message to self) sebagai bentuk interaksi yang terjadi selama sistem bekerja. Dimulai dari user mengakses halaman Question Collection dan diakhiri oleh report sebagai tanda berakhirnya proses bisnis dari fitur Tambah Soal.

# BAB IV
# USECASE DIAGRAM

## 1. Pengertian Use Case Diagram

**Use Case Diagram adalah** pemodelan untuk menggambarkan behavior / kelakuan sistem yang akan dibuat. Use case diagram menggambarkan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara sederhana, diagram use case digunakan untuk memahami fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang dapat menggunakan fungsi-fungsi tersebut.

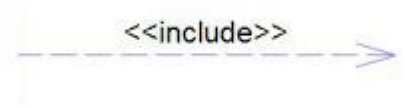
Menurut Rosa dan Salahudin use case digram tidak menjelaskan secara detail tentang penggunaan tiap use case, namun hanya memberi gambaran singkat hubungan antara use case, aktor, dan sistem. Melalui use case diagram kita dapat mengetahui fungsi-fungsi apa saja yang ada pada sistem (Rosa-Salahudin, 2011: 130).

Adapun syarat penamaan pada use case digram sendiri adalah nama didefinisikan sesederhana mungkin sehingga bisa dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case.

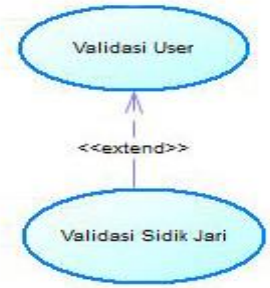
- Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor
- Aktor adalah orang atau system lain yang berinteraksi dengan system yang akan dibuat, jadi meskipun simbol dari aktor adalah gambar orang tapi aktor belum tentu merupakan orang

## 2. Simbol-Simbol Use Case Diagram

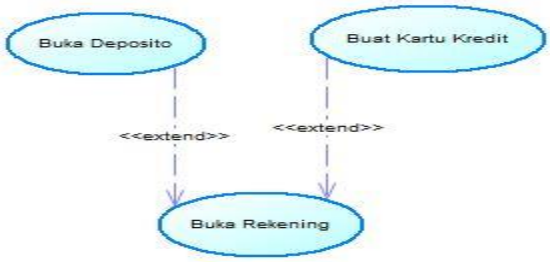
| Simbol | Deskripsi |
|---|---|
| Use Case   | Use case adalah fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau actor. biasanya use case diberikan penamaan dengan menggunakan kata kerja di awal frase nama use case |

| | |
|---|---|
| Aktor / actor<br>  <br> | Aktor adalah orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi meskipun simbol dari aktor ialah gambar orang, tapi aktor belum tentu merupakan orang. biasanya penamaan aktor dinamakan menggunakan kata benda di awal frase nama aktor |
| Asosiasi / association<br> | Asosiasi adalah komunikasi antara aktor dan use case yang berpartisipasi pada use case diagram atau use case yang memiliki interaksi dengan aktor. Asosiasi merupakan simbol yang digunakan untuk menghubungkan link antar element. |
| Ekstend / extend<br>  <br> | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri meski tanpa use case tambahan itu<br>arah panah mengarah pada use case yang ditambahkan |
| Include<br>  <br> | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan membutuhkan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini arah panah include mengarah pada use case yang dipakai (dibutuhkan) atau mengarah pada use case tambahan. |
| Generalisasi / generalization<br>  | Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah use case dimana fungsi yang satu merupakan fungsi yang lebih umum dari lainnya<br>arah panah mengarah pada use case yang menjadi generalisasinya (umum) |

**Penjelasan Simbol Extend**

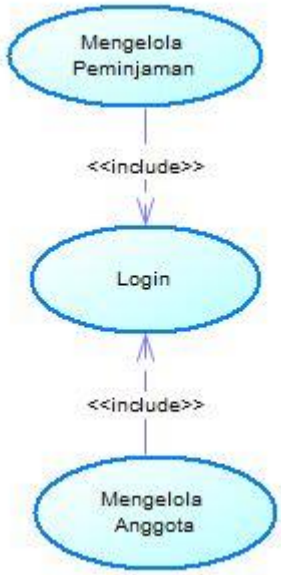



Pada gambar diatas use case Validasi User merupakan use case yang ditambahkan, dimana use case ini dapat berdiri sendiri tanpa use case tambahan (Validasi Sidik Jari). pada contoh diatas setelah pengguna melakukan validasi user, pengguna dapat mengembangkannya (opsional) dengan validasi sidik jari atau tidak.




Contoh lainnya adalah seperti pada gambar diatas. use case Buka Rekening merupakan use case yang ditambahkan sehingga use case ini dapat berdiri sendiri sedangkan use case Buka Deposito dan Buat Kartu Kredit merupakan use case tambahan yang berasal dari pengembangan use case extend. pada contoh diatas setelah pengguna melakukan Buka Rekening, pengguna dapat mengembangkannya / melanjutkannya (opsional) dengan Buka Deposito / Buat Kartu Kredit.

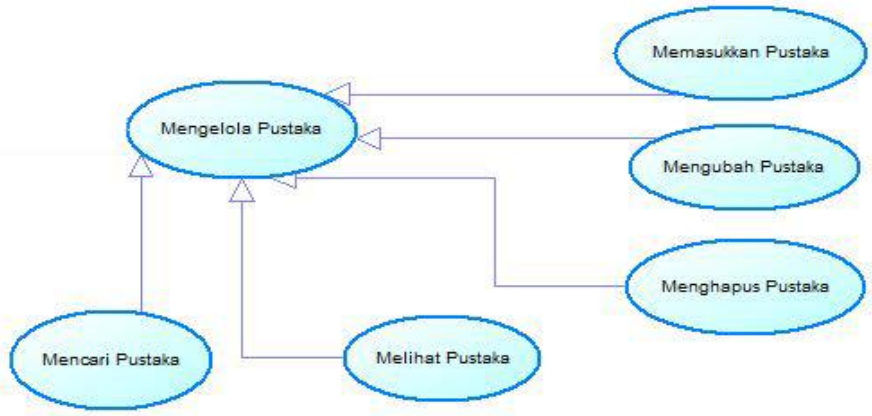
**Penjelasan Simbol Include**




Pada gambar diatas Use Case Login merupakan syarat / selalu dipanggil terlebih dahulu sebelum dijalankannya use case Mengelola Anggota atau use case Mengelola Peminjaman. Intinya perbedaan mendasar dari use case extend dan use case include adalah : use case extend digunakan untuk mengembangkan sebuah use case (use case inti) misalnya setelah melakukan Buka Rekening selanjutnya bisa melakukan apa lagi ?, dimana pada hubungan extend arah panah mengarah pada use case inti (use case ditambahkan). sedangkan use case include digunakan untuk menjelasakan bahwa sebuah use case memiliki sebuah syarat agar / ketentuan sebelum bisa dijalankan, misalnya saat kita akan mengelola anggota maka kita diwajibkan login terlebih dahulu. pada hubungan include arah panah mengarah pada use case tambahan (use case yang dipakai / dibutuhkan). Untuk semakin memperjelas, perhatikan contoh dibawah ini:

Selamat datang di Materi Dose. Pada kesempatan kali ini kita bakal mengupas tuntas tentang

**Penjelasan Simbol Generalisasi**




### 3. Use Case Skenario

Setiap use case diagram dilengkapi dengan skenario, skenario use case / use case skenario adalah alur jalannya proses use case dari sisi aktor dan system. Berikut adalah format tabel skenario use case.

| Nama Aktor | Reaksi Sistem |
|---|---|
| Skenario Normal | |
| Skenario Alternatif | |

Skenario use case dibuat per use case terkecil, misalkan untuk generalisasi maka scenario yang dibuat adalah use case yang lebih khusus. Skenario normal adalah scenario bila system berjalan normal tanpa terjadi kesalahan atau error. Sedangkan skenario alternatif adalah scenario bila system tidak berjalan normal atau mengalami error. Skenario normal dan skenario alternatif dapat berjumlah lebih dari satu. Alur skenario inilah yang nantinya menjadi landasan pembuatan sequence diagram / diagram sekuen

**Menentukan Aktor pada Use Case Diagram**

Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu (Kurt Bittner, Ian Spence. 2002). Cara termudah untuk menemukan aktor adalah dengan bertanya "SIAPA yang akan menggunakan sistem ?"

Namun tidak semua Aktor adalah manusia, aktor juga dapat berupa sistem lain (yang berada diluar sistem yang akan dibuat), ciri system sebagai actor adalah sebagai berikut:

- Jika system yang akan dibuat / dimodelkan bergantung pada sistem lain untuk melakukan sesuatu, maka sistem lain itu adalah aktor.
- Jika sistem yang akan dibuat / dimodelkan meminta (request) informasi dari sistem lain, maka sistem lain itu adalah aktor

Untuk kasus sistem lain yang bertindak sebagai aktor, dapat di ilustrasikan sebagai berikut : misalkan sebuah Sistem Akademik baru dapat menampilkan nilai mahasiswa apabila pembayaran mahasiswa sudah lunas, artinya system akademik memerlukan info dari sistem pembayaran. maka saat kita akan memodelkan use case diagram Sistem Akademik, kita akan memasukkan sistem pembayaran sebagai aktor.

**Menentukan Use Case pada Use Case Diagram**

Sebuah use case harus mendeskripsikan sebuah pekerjaan dimana pekerjaan tersebut akan memberikan NILAI yang bermanfaat bagi aktor (Kurt Bittner, Ian Spence. 2002).

Untuk menemukan use cases, mulailah dari sudut pandang aktor, misalnya dengan bertanya:

1. Informasi apa sajakah yang akan didapatkan aktor dari sistem ?
2. Apakah ada kejadian dari sistem yang perlu diberitahukan ke aktor ?

Sedangkan dari sudut pandang sistem, misalnya dengan pertanyaan sebagai berikut:

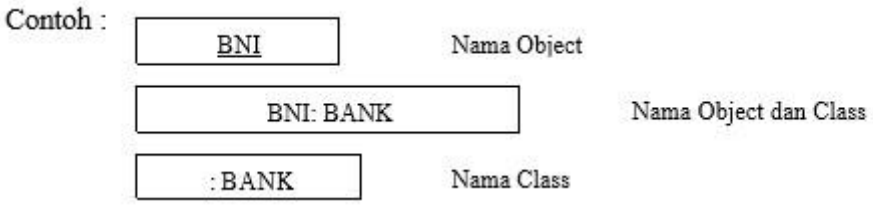
1. Apakah ada informasi yang perlu disimpan atau diambil dari sistem ?
2. Apakah ada informasi yang harus dimasukkan oleh aktor?

Pengertian ini penting untuk diingat, karena dari hal inilah akan menentukan bahwa sebuah use case tidak akan menjadi terlalu kecil. Karena use case yang terlalu kecil tidak akan memberikan nilai bagi aktor.

# BAB V

# SEQUENCE DIAGRAM

**1. Pengertian Sequence Diagram** adalah salah satu dari diagram - diagram yang ada pada UML, sequence diagram ini adalah diagram yang menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object.* Sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem. Dalam UML, *object* pada *sequence diagram* digambarkan dengan segi empat yang berisi nama dari *object* yang digarisbawahi. Pada *object* terdapat 3 cara untuk menamainya yaitu : nama *object*, nama *object* dan *class,* dan nama *class*. Berikut contoh dari ketiga cara tersebut :



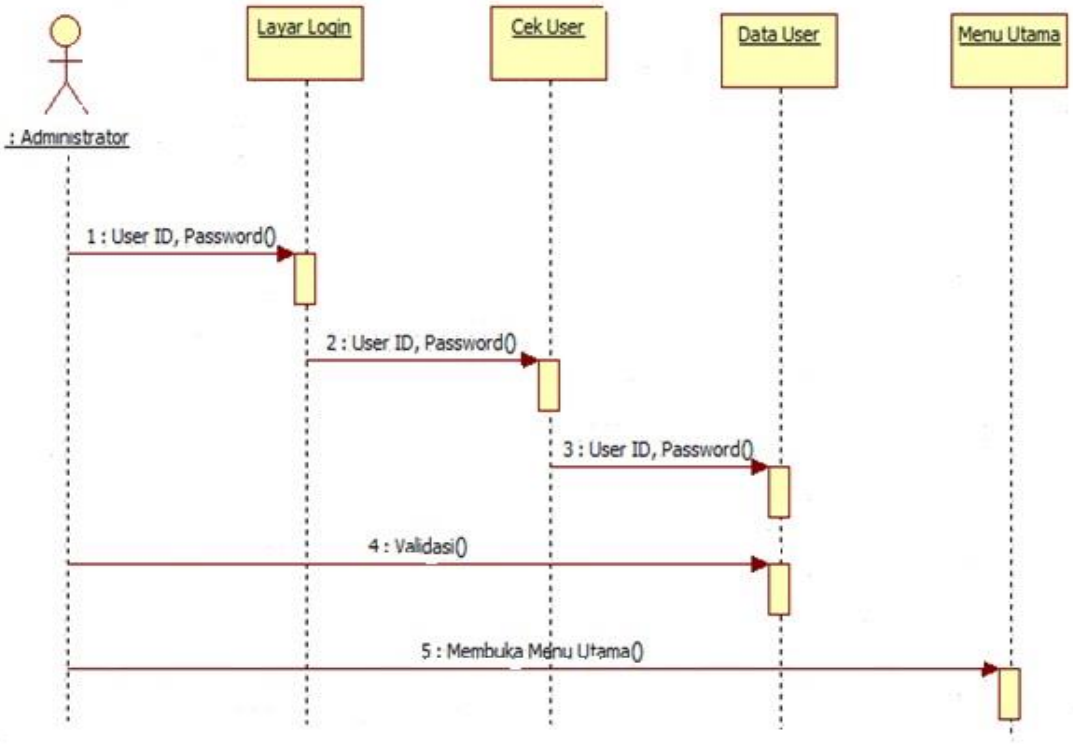

Dalam sequence diagram, setiap *object* hanya memiliki garis yang digambarkan garis putus-putus ke bawah. Pesan antar *object* digambarkan dengan anak panah dari *object* yang mengirimkan pesan ke *object* yang menerima pesan.

**2. Komponen - komponen**

Berikut komponen - komponen yang ada pada sequence diagram :

- Object - adalah komponen berbentuk kotak yang mewakili sebuah class atau object. Mereka mendemonstrasikan bagaimana sebuah object berperilaku pada sebuah system.
- Activation boxes - adalah komponen yang berbentuk persegi panjang yang menggambarkan waktu yang diperlukan sebuah object untuk menyelesaikan tugas. Lebih lama waktu yang diperlukan, maka activation boxes akan lebih panjang.
- Actors - adalah komponen yang berbentuk *stick figure.* Komponen yang mewakili seorang pengguna yang berinteraksi dengan syst
- Lifeline - adalah komponen yang berbentuk garis putus - putus. Lifeline biasanya memuat kotak yang berisi nama dari sebuah object. Berfungsi menggambarkan aktifitas dari object

Berikut merupakan contoh sederhana dari Sequence Diagram :




Gbr 8. Contoh Diagram Sequence

## Penjelasan

Pada Sequence Diagram diatas, bisa dilihat bahwa yang menjadi Actors adalah Administrator. Activation boxes biasanya memilik garis yang memberitahu aktifitas yang terjadi ketika actors atau objects berinteraksi ke object lain.

# BAB VI

# CLASS DIAGRAM

## 1. Pengertian Class Diagram

**Class diagram adalah** visual dari struktur sistem program pada jenis-jenis yang di bentuk. Class diagram merupakan alur jalannya database pada sebuah sistem.
Class diagram merupakan penjelasan proses database dalam suatu program. Dalam sebuah laporan sistem maka class diagram ini wajib ada.

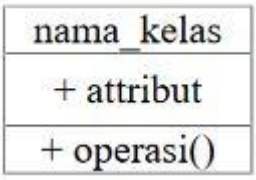
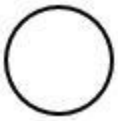
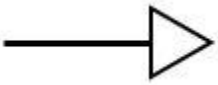
**Menurut Para ahli** Satzinger (2011:28) Diagram kelas atau class diagram menjelaskan struktur sistem dari segi pendefinisian class-class yang akan dibuat untuk membangun sebuah sistem.

## 2. Fungsi atau Manfaat Class Diagram

Adapun fungsi dan manfaat dari class diagram adalah sebagai berikut.

- Menjelaskan suatu model data untuk program informasi, tidak peduli apakah model data tersebut sederhana maupun kompleks.
- Dengan menguasai class diagram maka akan meningkatkan pemahaman mengenai gambaran umum skema dari suatu program.
- Mampu menyatakan secara visual akan kebutuhan spesifik suatu informasi serta dapat berbagi informasi tersebut ke seluruh bisnis.
- Dengan Class Diagram dapat dibuat bagan secara terperinci dan jelas, dengan cara memperhatikan kode spesifik apa saja yang dibutuhkan oleh program. Hal ini mampu mengimplementasikan ke struktur yang dijelaskan.
- Class Diagram mampu memberikan penjelasan implementasi-independen dari suatu jenis program yang digunakan, kemudian dilewatkan diantara berbagai komponennya.

### 3. Simbol-simbol Class diagram

| Simbol | Nama | Keterangan |
|---|---|---|
|  | Kelas | Kelas pada struktur sistem. |
|  | Interface | Sama dengan konsep interface dalam pemrograman berorientasi objek. |
|  | Association | Relasi antarclass dengan arti umum, asosiasi biasanya juga disertai dengan Multiplicity. |
|  | Directed Association | Relasi antarkelas dengan makna kelas yang atau digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity. |
|  | Generalisasi | Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus). |
|  | Dependency | Relasi antarkelas dengan makna kebergantungan antarkelas’ |
|  | Aggregation | Relasi antarkelas dengan makna semua-bagian (whole-part) |

## Contoh Class Diagram


# DAFTAR PUSTAKA


1. Booch, B. G., Rumbaugh, J., & Jacobson, I. (2005). *The Unified Modeling Language User Guide*.
2. Dennis, A., Wixom, B. H., & Tegarden, D. (2012). *Systems Analysis and Design with UML*.
3. Fowler, M. (2004). *UML Distilled Third Edition A Brief Guide to The Standard Object Modeling Language*. Boston: Pearson Education, Inc.
4. Martin, R. C., Cecil, R., & Martin, R. C. (2002). *UML for Java Programmers*. New Jersey: Alan Apt
5. Rumbaugh, J., Jacobson, I., & Booch, G. (2005). *The Unified Modeling Language Reference Manual, Second Edition*. Boston: Pearson Education, Inc.