BAB 2

LANDASAN TEORI

2.1 *Artificial Intellegence* (AI)

Artificial Intellegence atau AI dalam bahasa Indonesia artinya kecerdasan buatan yaitu, kecerdasan buatan yang ditunjukan oleh suatu entitas ilmiah. Kecerdasan dibuat untuk dimasukkan ke dalam suatu mesin/ komputer supaya bisa melakukan pekerjaan seperti yang biasa dikerjakan manusia [8]. Karena itu kecerdasan buatan harus didasarkan pada prinsip-prinsip teoretikal dan terapan yang menyangkut struktur data yang digunakan dalam representasi pengetahuan (*knowledge representation*), algoritma yang diperlukan dalam penerapan pengetahuan itu, serta teknik-teknik bahasa dan pemrograman yang dipakai dalam implementasinya.

Beberapa persoalan yang ditangani oleh kecerdasan buatan antara lain adalah permainan (*game*), pembuktian teorema, pemecahan problema umum, persepsi (*vision* dan percakapan), pemahaman bahasa alamiah, pemecahan problema pakar atau *expert* (matematika simbolik, diagnosis medis, rancang bangun rekayasa, analisis kimia). Hal – hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah dikehidupan yang nyata [8].

2.2 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah salah satu cabang dari ilmu komputer, *Artificial Intellegence* dan *Computational Linguistik* yang berkonsentrasi pada hubungan komputer dengan bahasa alami manusia atau bisa disimpulkan bahwa NLP membahas terkait interaksi manusia dengan komputer. Banyak hal yang menjadi kajian dalam NLP diantaranya adalah *Natural Language Understanding*, Membuat komputer mengerti bahasa alami manusia yang diinputkan dan melibatkan generasi NLP selanjutnya.

Tahap – tahap dalam *Natural Language Processing* menurut para ahli: Menurut Rich dan Knight (1991,pp379-380) tahapan *Natural Language Processing* terdiri dari beberapa level analisis. Level analisis tersebut adalah:

a. *Morphological Analysis*

Kata – kata secara indvidu dilakukan berdasarkan komponennya, dan token yang tidak termasuk, seperti tanda baca dipisahkan dari kata – kata yang ada. Analisis ini memperhatikan arti dari setiap komponen yang ada membentuk suatu kata. Analisis morfologi sangat penting untuk menetukan aturan kata yang ada dalam pada kalimat, termasuk tata bahasa.

b. *Syntax Analysis*

Urutan – urutan linear dari kata – kata diubah menjadi struktur yang menunjukan bagaimana satu kata berhubungan dengan kata yang lain. Dimana analisis ini mempelajari aturan untuk menggabungkan kata dari frase dan kalimat, serta menggunakan aturan tersebut untuk menguraikan dan membentuk kalimat. Level analisis ini yang lebih banyak berhasil.

c. *Semantic Analysis*

Struktur yang diciptakan dari hasil analisis sintaksis akan diperiksa arti yang sebenarnya. Kalau struktur tersebut tidak memenuhi persyaratan kaidah bahasa, maka kalimat tersebut dapat dianggap anomali semantik.

d. *Discourse Integration*

Mungkin saja arti dari suatu kalimat bergantung dari kalimat sebelumnya dan mungkin dapat mempengaruhi kalimat – kalimat selanjutnya.

2.3 *Structure Query Language* (SQL)

Structure Query Language atau SQL merupakan suatu bahasa yang digunakan untuk mengakses *database.* SQL juga sering disebut *query*. Secara umum SQL terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML) [6].

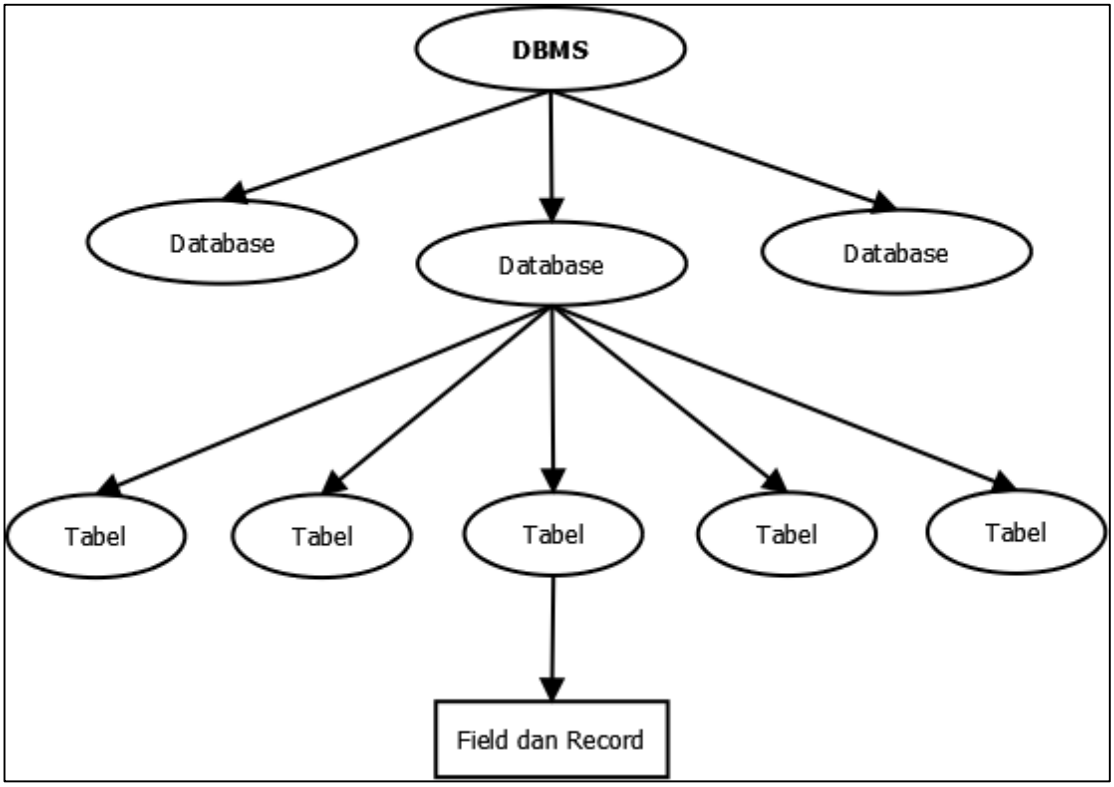
DDL digunakan untuk mendefinisikan, mengubah, serta menghapus *database* dan objek – objek yang diperlukan *database*. Secara umum, DDL yang digunakan adalah *Create, Alter, Use, Drop* yang biasanya digunakan oleh administrator dalam pembuatan suatu aplikasi *database*. DML digunakan untuk memanipulasi data yang ada dalam suatu tabel. Perintah yang dapat dilakukan adalah *Select, Insert, Update, Delete* [6].

Fungsi Agregat, yaitu fungsi khusus yang melibatkan sekelompok data, secara umum fungsi agregat adalah SUM, COUNT, AVG, MAX, dan MIN. Fungsi ini digunakan pada bagian *select*, syarat untuk fungsi ini adalah diletakkan pada bagian *having*, bukan *where*. *Sub query* adalah suatu *query* yang sudah menjadi kompleks terutama melibatkan lebih dari satu tabel atau fungsi agregat [6].

2.4 *Database*

Database atau basis data adalah kumpulan data yang disimpan secara sistematis di komputer dan dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur, dan juga batasan – batasan dan yang akan disimpan. Basis data merupakan aspek yang sangat penting dalam sistem informasi dimana basis data merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat menghindari duplikasi data, hubungan antar data yang tidak jelas, organisasi data, dan juga ubah data yang rumit. Dalam aplikasi yang penulis buat, maka penulis menggunakan aplikasi dalam memanajemen *database* yaitu MySQL [6].

Dalam konsep *database*, urutan atau hierarki *database* sangatlah penting. Urutan atau hierarki *database* digambarkan dalam gambar sebagai berikut :



Gambar 2. 1 Hierarki Database

2.5 *Scanner*

Analisis Leksikal (*Scanner*) merupakan antarmuka antara kode program sumber dan analisa sintaktik (parser). Atau dalam pengertiannya adalah sebuah proses yang mendahului parsing sebuah rangkaian karakter. Scanner melakukan pemeriksaan karakter per karakter pada teksmasukan, memecah sumber program menjadi bagian-bagian disebut Token. Proses parsing akan lebih mudah dilakukan bila inputnya sudah berupa token [5].

Analisis leksikal membuat pekerjaan membuat sebuah parser jadi lebih mudah daripada membangun nama setiap fungsi dan variabel darikarakter-

karakter yang menyusunnya, dengan analisis leksikal parser cukup hanya berurusan dengan sekumpulan token dan ni ai sintaksis masing-masing.

Terlepas dari efisiensi pemrograman yang dapat dicapai dengan penggunaannya, proses kerja analisis leksikal yang membaca lebih dari sekali setiap karakter dari input yang diberikan menjadikan penganalisa leksikal sebagai sub-sistem yang paling intensif melakukan komputasi, terutama bila digunakan dalam sebuah kompilator.

Kompilator adalah sebuah program yang membaca suatu program yang ditulis dalam suatu bahasa sumber (source language) dan menterjemahkannya ke dalam suatu bahasa sasaran (target language). Dalam penguraian struktur kalimat, penganalisa leksikal menganalisa setiap kata dalam kalimat, kemudian menentukan jenis kelas katanya.

Hasil dari penganalisa leksikal ini digunakan oleh penganalisa sintaks yang akan memeriksa urutan simbol-simbol kelas kata tersebut dalam kalimat. Analisa kata dalam kalimat ini dilakukan oleh penganalisa leksikal berdasarkan kecocokan kata dengan aturan-aturan leksikal berupa ekspresi regular yang sudah didefinisikan.

Tugas *scanner* bisa dirangkum sebagai berikut :

1. Melakukan pembacaan kode sumber dengan menurut karakter demi karakter.
2. Mengenali besaran leksik.
3. Mentransformasi menjadi sebuah token dan menentukan jenis token nya.
4. Mengirimkan token.
5. Menangani kesalahan.
6. Menangani tabel simbol.

Didalam aplikasi NLP sistem cerdas yang akan dibuat, yang dimaksud dengan program sumber yang diolah oleh scanner adalah berupa kalimat input dari pengguna.

Ketika scanner menerima input berupa stream karakter kemudian memilah menjadi satuan leksik, satuan leksik tersebut terdiri atas simbolsimbol satuan yang jika dikombinasikan akan mempunyai arti yang berbedabeda. Simbol-simbol yang bisa dipergunakan dalam sebuah bahasa tentunya terbatas jumlahnya, yang membentuk sebuah himpunan dan disebut sebagai abjad (alphabet).

Tata bahasa (grammatika) adalah sekumpulan dari himpunan variabel-variabel, simbol-simbol terminal, simbol non-terminal, simbol awal yang dibatasi oleh aturan-aturan produksi. Aturan produksi adalah pusat dari tata bahasa yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi suatu *string* ke bentuk lainnya.

Dalam pembicaraan grammar, anggota alfabet dinamakan simbol terminal atau token. Kalimat adalah string yang tersusun atas simbol-simbol terminal. Bahasa adalah himpunan kalimat-kalimat. Anggota bahasa bisa berupa tak berhingga hingga kalimat.

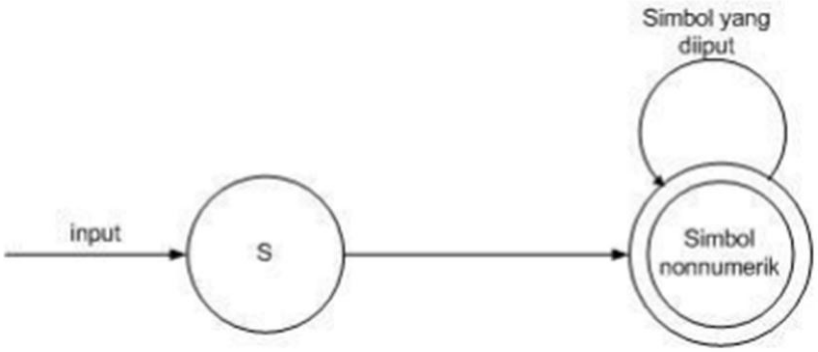
Simbol-simbol berikut adalah symbol terminal :

1. Huruf kecil alphabet, misalnya :a, b, c
2. Simbol operator, misalnya : +, -, dan ‘
3. Simbol tanda baca, misalnya : (,), dan ;

Sedangkan simbol-simbol berikut adalah simbol non terminal :

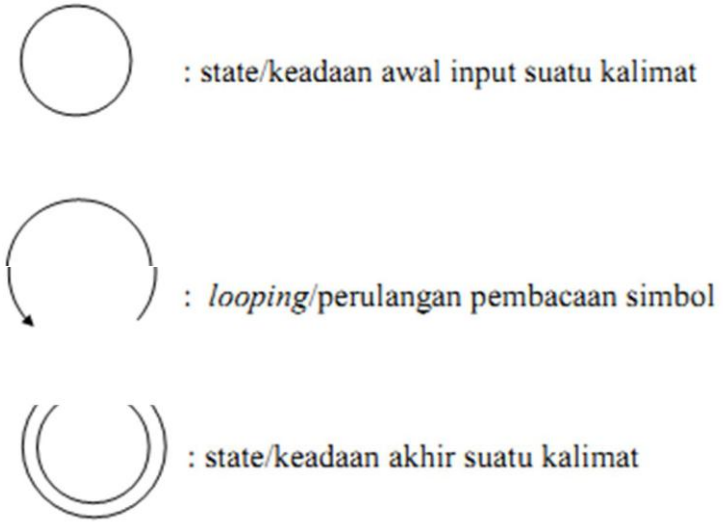
1. Huruf besar awal alphabet, misalnya : A, B, C
2. Huruf S sebagai simbol awal

Pada saat scanner membaca input, tools yang digunakan untuk menggambarkan perpindahan dari posisi satu ke posisi lainnya adalah diagram transisi. Diagram transisi dapat dilihat pada gambar berikut ini :



Gambar 2. 2 Diagram Transisi

Keterangan :



Gambar 2. 3 Keterangan diagram Transisi

2.5.1 Besaran Leksik

Besaran – besaran yang membangun bahasa leksik meliputi hal – hal sebagai berikut :

2.5.1 Identifier

Identifier bisa berupa *keyword* atau nama. *Keywords* adalah kata kunci yang sudah didefinisikan oleh suatu bahasa seperti SELECT, FROM, WHERE, ALTER, dan SQL.

2.5.2 Nilai Konstanta

Nilai Konstanta adalah suatu konstanta yang terdapat pada program. Bisa berupa konstanta *integer, real, boolean, character, string,* dan sebagainya

2.5.3 Operator dan Delimiter

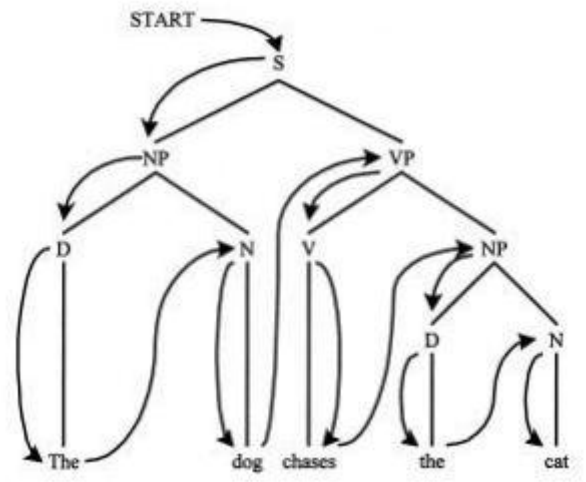
Operator adalah suatu operator matematika seperti (+,-,*,/), ataupun operator logika seperti (>,<,=). Delimiter berguna sebagai pemisah atau pembatas, seperti : , ( ), ;, *White Space. White Space* adalah pemisah yang diabaikan program, seperti spasi, karakter Enter, Ganti baris, akhir file.

2.6 *Parser*

Parser merupakan tahapan analisis sintaksis yang berguna untuk memeriksa urutan kemunculan *token.* Didalam mengimplementasikan sebuah metode *parsing* kedalam program diperlukan tiga hal sebagai berikut, rentang waktu eksekusi, penanganan kesalahan, penanganan kode [5]. Metode *parsing* bisa digolongkan sebagai berikut :

1. *Top Down*

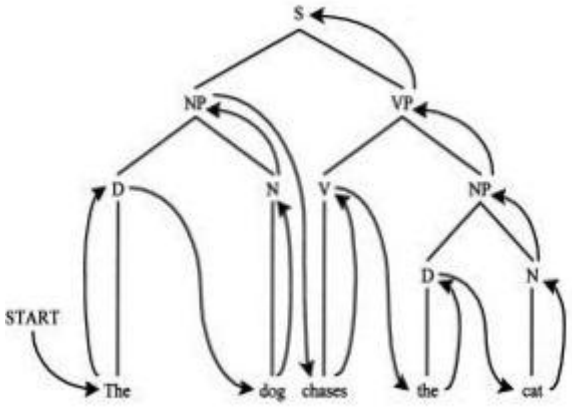
   Kalau dilihat dari terminologi pohon penurunan, metode ini melakukan penelusuran dari *root/*puncak menuju *leaf/*daun. Metode *top down* sendiri meliputi, *Backtrack / backup* : *Brute force, No Backtrack* : *Recursive Descent Parser* [5]. Dapat dilihat pada Gambar 2.4 berikut :



Gambar 2. 4 Metode Top Down

2. *Bottom Up*

   Metode ini melakukan penelusuran dari *leaf*/daun menuju *root*/puncak. Dapat dilihat pada Gambar 2.5 berikut ini :



Gambar 2. 5 Metode Bottom Up

2.7 *Unified Modelling Language* (UML)

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh *Grady Booch, Jim Rumbaugh,* dan *Ivar Jacobson*.

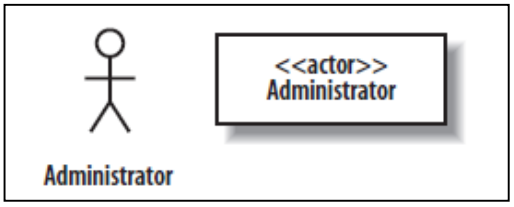
Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem. Proses pemodelan data menggambarkan keseluruhan proses bisnis yang akan dilakukan oleh sebuah sistem informasi. Proses model juga menjelaskan data-data yang terlibat dalam proses tersebut. Salah satu pemodelan data adalah dengan UML [7].

2.7.1 *Use case* Diagram

Use case adalah kasus atau situasi dimana sistem digunakan untuk memenuhi salah satu atau lebih dari kebutuhan pengguna. *Use case* menangkap sepotong fungsionalitas yang sistem sediakan. *Use case* merupakan jantung dari sebuah model, karena *use case* mempengaruhi dan membimbing semua elemen lain dalam sistem [7].

a. Aktor

Aktor digambarkan dalam UML, menggunakan notasi *stick man* atau *stereotyped box* dan berlabel dengan nama yang tepat seperti gambar berikut [7].



Gambar 2. 6 Aktor

b. *Use case*

Sebuah *use case* atau pekerjaan, dapat sesimpel memperbolehkan user untuk log in atau dapat serumit mengeksekusi transaksi database global yang terdistribusi.

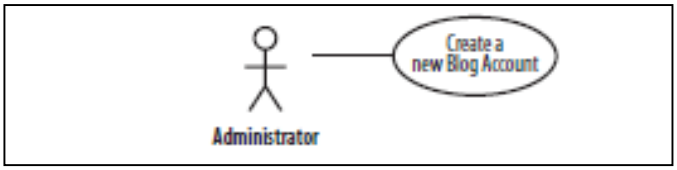
Gambar dibawah ini menunjukan bagaimana interaksi ditangkap menggunakan *use case* [7].



Gambar 2. 7 *Use case*

c. *Communication line*

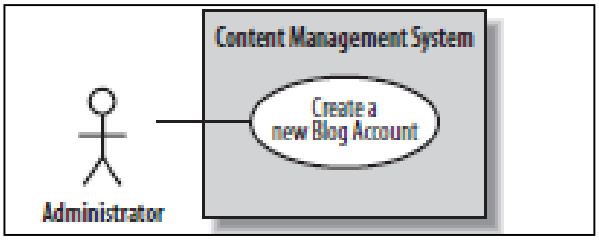
Sebuah *communication line* menghubungkan actor dan *use case* untuk menunjukan actor yang berpartisipasi dalam *use case*. Sebagai contoh, aktor Administrator terlibat dalam *use case Create a new Blog Account* [7].



Gambar 2. 8 *Communication Line*

d. *System Boundaries*

Untuk menampilkan *system's boundary* pada use case diagram, gambar sebuah kotak disekitar semua use case tapi aktor tetap berada diluar kotak [7].



Gambar 2. 9 *System Boundaries*

2.7.2 *Activity* Diagram

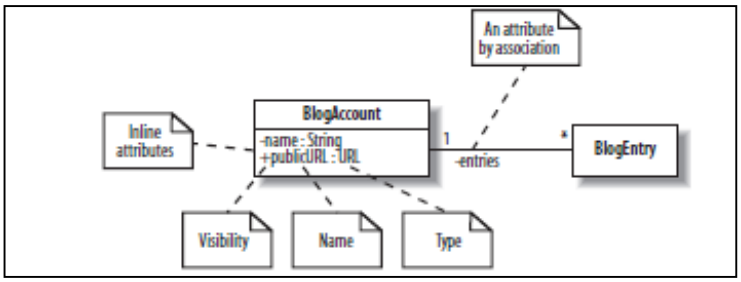
Use case menggambarkan perilaku sistem. Kelas menggambarkan berbagai jenis benda-benda yang dibutuhkan dalam sistem. Diagram aktifitas

menunjukan proses bisnis dari sistem yang sedang berjalan. Diagram aktifitas berupa urutan proses dari proses bisnis yang ada [7].

2.7.3 *Class* Diagram

a. Atribut

Sebuah *class's attributes* adalah potongan informasi yang mewakili keadaan suatu objek. Atribut ini dapat direpresentasikan pada digram kelas baik dengan menempatkan mereka didalam bagian dari kotak atau oleh asosiasi dengan kelas lain, seperti yang ditunjukan pada gambar berikut [7].



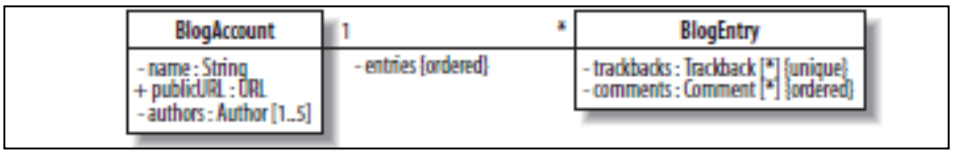
Gambar 2. 10 Atribut

b. Nama dan Tipe

Dalam atribut nama dapat menjadi bagian dari karakter, tapi tidak boleh ada dua atribut dalam kelas yang sama dan mempunyai nama yang sama. Jenis atribut dapat bervariasi tergantung bagaimana kelas akan diimplementasikan dalam sistem [7].

c. Multiplicity

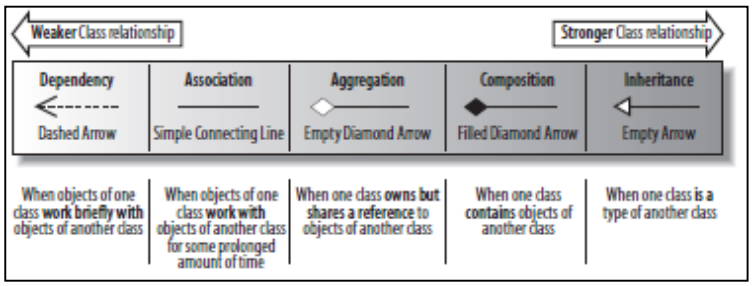
Terkadang atribut akan mewakili lebih dari satu objek. Bahkan atribut bisa mewakili sejumlah objek, seperti mendeklarasikan bahwa atribut adalah array. *Multiplicity* memungkinkan untuk menentukan bahwa atribut sebenarnya merupakan koleksi benda-benda dan dapat diterapkan baik *inline* dan atribut oleh asosiasi, seperti ditunjukan pada gambar berikut [7].



Gambar 2. 11 *Multiplicity*

d. Class Relationship

Kelas tidak hidup diruang hampa, mereka bekerja bersama-sama menggunakan berbagai jenis hubungan. Hubungan antar kelas datang dalam kekuatan yang berbeda, seperti ditunjukkan pada gambar dibawah ini [7].

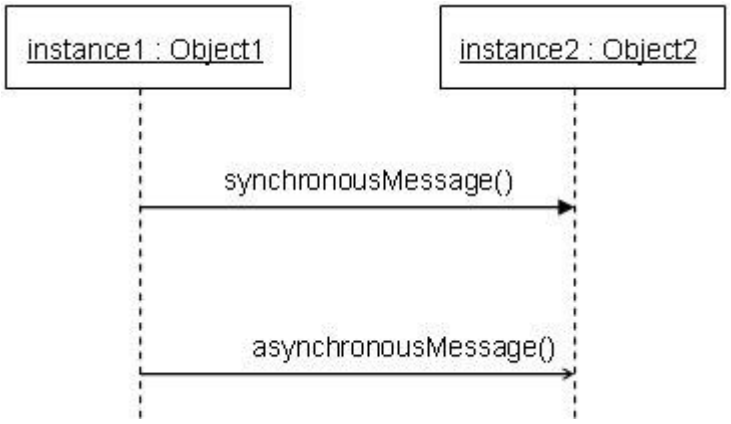


Gambar 2. 12 Class Relationship

2.7.4 *Sequence* Diagram

Sequence Diagram menggambarkan interaksi antar objek didalam dan disekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian lengkah-langkah yang dilakukan sebagai respon dari event untuk menghasilkan *output* tertentu. Diawali dari pada men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internandal *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari suatu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metode dari *class*. *Activation bar* menunjukan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*. Untuk objek-objek yang memiliki

sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller*, dan *presistent entity* [7].



Gambar 2. 13 *Sequence* Diagram

2.8 *Object Oriented Programming*

Pemrograman berorientasi objek (OOP) merupakan paradigma pemrograman yang menganggap semua benda adalah objek, yaitu menganggap segala sesuatu dalam bahasa pemrograman adalah objek, bahkan variabel pun dianggap sebahai suatu objek. Konsep dasar dari pemrograman berorientasi objek menekankan kepada konsep: Objek (*Object*)"Objek merupakan kumpulan variabel dan fungsi yang saling berkaitan".

1. Kelas (*Class*) "*Class* merupakan suatu rancangan dari objek itu sendiri. Jadi satu *class* dapat berisi banyak objek".
2. Pembungkusan (*Encapsulation*) " *Encapsulation* merupakan cara menyembunyikan detail informasi dan fungsionalitas yang ada pada suatu kelas".
3. Pewarisan (*Inheritance*) "*Inheritance* merupakan kemampuan untuk meng-*extend class* yang akan mewarisi semua karakter dari partner *class*-nya".
4. Polimorfisme (*Pholymorphism*) " *Pholymorphism* merupakan kemampuan objek-objek yang berada dikelas, tetapi terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama".

2.9 *Tools* Pembangunan Sistem

2.9.1 Java

Java adalah bahasa pemrograman yang dapat dijalankan diberbagai komputer. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung dengan Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++, Aplikasi-aplikasi berbasis java umumnya dikompilasi kedalam p-code (*bytecode*) dan dapat dijalankan pada berbagai mesin virtual java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non spesifik (*general purpose*), dan secara khusus didesain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan dibeberapa platform sistem operasi berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan dimana pun". Saat ini java merupakan bahasa pemrograman yang populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak ataupun aplikasi [9].

2.9.2 Eclipse

Eclipse adalah sebbuah IDE (*Integereted Bevelopment Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan disemua platform. Berikut adalah sifat dari Eclipse; Multi-platform, Multi-language, Multi-role. Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *Open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini [9].

2.9.3 MySQL

MySQL adalah sebuah sistem manajemen *database* relasi (*relation database management system*) yang bersifat *open source*. MySQL merupakan buah pikiran dari Michael "Monty" Widenius, Davied axmark dan Allan Larson yang dimulai tahun 1995. Mereka bertiga kemudian mendirikan perusahaan bernama MySQL AB di swedia. MySQL merupakan aplikasi *database* server. Perkembangannya disebut SQL yang merupakan kepanjangan dari *Structured*

Query Language. SQL merupakan bahasa terstruktur yang digunakan untuk mengolah *database* beserta isinya. Kita dapat memanfaatkan MySQL untuk menambah, mengubah, dan menghapus data yang berada dalam *database* [6]. MySQL versi 1.0 rilis pada Mei 1996 dan penggunaannya hanya terbatas dikalangan perusahaan saja. Barulah pada bulan Oktober 1996, MySQL versi 3.11.0 dirilis ke masyarakat luas. MySQL menggunakan bahasa standar SQL (*Structure Query Language*) sebagai bahasa interaktif dalam mengelola data. MySQL memiliki kinerja, kecepatan, proses dan ketangguhan yang tidak kalah dibanding *database* manajemen *system* lain nya seperti Oracle, Sysbase, Unify, dan sebagainya. MySQL dapat berjalan diatas banyak sistem operasi Linux, Windows, Solaris, FreeBSD, Mac OS X dan lain sebagainya [6].