

BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar

Sistem pakar menirukan perilaku seorang pakar dalam menangani suatu persoalan. Pada suatu kasus seorang pasien mendatangi dokter untuk memeriksa badannya yang mengalami gangguan kesehatan, maka dokter atau pakar kesehatan akan memeriksa dan melakukan diagnosa. Bila dokter cukup sibuk dan

Pelaksana diagnosa digantikan oleh sebuah sistem pakar, maka sistem pakar diharapkan dapat membantu memahami dan menganalisa keadaan pasien dan menemukan penyakit yang diderita pasien itu. Sistem pakar diharapkan juga untuk menghasilkan dugaan atau hasil diagnosa yang sama dengan diagnosa yang dilakukan oleh seorang ahli. Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar-pakar yang ahli di bidangnya (Andri Saputra, 2011 : 203).

II.2.1. Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar mengandung : keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan. Keahlian adalah suatu kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian adalah :

1. Fakta-fakta pada lingkup permasalahan tertentu.
2. Teori-teori pada lingkup permasalahan tertentu.
3. Prosedur-prosedur dan aturan-aturan berkenaan dengan lingkup permasalahan tertentu.
4. Strategi-strategi global untuk menyelesaikan masalah.
5. *Meta-knowledge* (pengetahuan tentang pengetahuan).

II.1.2. Aktivitas Sistem Pakar

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan ahli. Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan (domain), menyusun kembali pengetahuan jika dipandang perlu, memecah aturan-aturan jika dibutuhkan, dan menentukan relevan tidaknya keahlian mereka. Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu :

1. Tambahan pengetahuan (dari para ahli atau sumber-sumber lainnya).
2. Representasi pengetahuan (ke komputer).
3. Inferensi pengetahuan.
4. Pengalihan pengetahuan ke user.

II.I.3. Bentuk Sistem Pakar

Pengetahuan yang disimpan di komputer disebut dengan nama basis pengetahuan. Ada 2 tipe pengetahuan, yaitu : fakta dan prosedur (biasanya berupa aturan). Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basisdata, maka komputer harus dapat diprogram untuk membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi (inference engine). Sebagian besar sistem pakar komersial dibuat dalam bentuk rule-based systems, yang mana pengetahuannya disimpan dalam bentuk aturan-aturan. Aturan tersebut biasanya berbentuk IF-THEN. Fitur lainnya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan inilah yang membedakan sistem pakar dengan sistem konvensional. Ada 4 bentuk sistem pakar, yaitu :

1. Berdiri sendiri. Sistem pakar jenis ini merupakan software yang berdiri-sendiri tidak tergantung dengan software yang lainnya.
2. Tergabung. Sistem pakar jenis ini merupakan bagian program yang terkandung didalam suatu algoritma (konvensional), atau merupakan program dimana didalamnya memanggil algoritma subrutin lain (konvensional).
3. Menghubungkan ke software lain . Bentuk ini biasanya merupakan sistem pakar yang menghubungkan ke suatu paket program tertentu, misalnya DBMS.
4. Sistem Mengabdikan. Sistem pakar merupakan bagian dari komputer khusus yang dihubungkan dengan suatu fungsi tertentu. Misalnya sistem pakar yang

digunakan untuk membantu menganalisis data radar (Andri Saputra, 2011 : 203).

II.2. Metode *Certainty Factor* (CF)

Certainty factor (CF) menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Faktor kepastian ini merupakan bentuk penggabungan kepercayaan dan ketidakpercayaan dalam suatu bilangan tunggal. *Certainty Theory* ini diusulkan oleh Shortliffe dan Buchanan pada tahun 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang pakar. Teori ini berkembang bersamaan dengan pembuatan sistem pakar MYCIN. Team pengembang MYCIN mencatat bahwa tim ahli sering kali menganalisa informasi yang ada dengan ungkapan seperti misalnya: mungkin, kemungkinan besar, hampir pasti. Untuk mengakomodasi hal ini tim MYCIN menggunakan *Certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi.

Beberapa *evidence* dapat dikombinasikan untuk menentukan CF dari suatu hipotesis. Untuk sistem ini, tingkat kepastian sistem terhadap kesimpulan yang diperoleh dihitung berdasarkan nilai probabilitas penyakit karena adanya *evident/gejala* tertentu. Faktor kepastian bukanlah suatu peluang, tetapi merupakan ukuran tingkat kepercayaan terhadap suatu *evidensi*. CF mempresentasikan tingkat kepercayaan bahwa suatu *evidensi* adalah benar.

Certainty Theory ini diusulkan oleh Shortliffe dan Buchanan pada tahun 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexact reasoning*) seorang

pakar. Teori ini berkembang bersamaan dengan pembuatan sistem pakar MYCIN. Team pengembang MYCIN mencatat bahwa tim ahli sering kali menganalisa informasi yang ada dengan ungkapan seperti misalnya: mungkin, kemungkinan besar, hampir pasti. Untuk mengakomodasi hal ini tim MYCIN menggunakan *certainty factor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi. Secara umum, rule direpresentasikan dalam bentuk sebagai berikut:

IF E1 [AND/OR] E2 [AND/OR] ... En

THEN H (CF = CF)

II.2.1. Model Perhitungan *Certainty Factor* dengan *Rule*

Certainty Factor (CF) menunjukkan ukuran kepastian terhadap suatu fakta atau aturan. Faktor kepastian ini merupakan bentuk penggabungan kepercayaan dan ketidakpercayaan dalam suatu bilangan tunggal. Berikut notasi faktor kepastian:

$$CF(P_k, G) = MB(P_k, G) - MD(P_k, G)$$

Beberapa *evidence* dapat dikombinasikan untuk menentukan CF dari suatu hipotesis. Untuk sistem ini, tingkat kepastian sistem terhadap kesimpulan yang diperoleh dihitung berdasarkan nilai probabilitas penyakit karena adanya evident/gejala tertentu [5]. Jika ada gejala dan penyakit sebagai *hypothesis* maka tingkat kepastian diformulasikan sebagai $CF(P_k, G)$:

$$MB(P_k, G) = \begin{cases} 1 & .P(P_k)=1 \\ \frac{\max[P(P_k|G), P(P_k)] - P(P_k)}{\max[1, 0] - P(P_k)} & .yanglain \end{cases}$$

$$MD(P_k, G) = \begin{cases} 1 & .P(P_k)=1 \\ \frac{\min[P(P_k|G), P(P_k)] - P(P_k)}{\min[1, 0] - P(P_k)} & .yanglain \end{cases}$$

Di mana:

$P(P_k)$: probabilitas kerusakan P_k

G : Gejala

CF : Certainty Factor (faktor Kepastian) dalam hipotesis H yang dipengaruhi oleh evidence (fakta) E .

MB : Measure of Belief (tingkat keyakinan), merupakan ukuran kepercayaan dari hipotesis H dipengaruhi oleh evidence (fakta) E .

MD : Measure of Disbelief (tingkat ketidakyakinan) merupakan ukuran ketidakpercayaan hipotesis H dipengaruhi oleh fakta E .

H : Hipotesa atau konklusi yang dihasilkan

Jika ada kaidah lain termasuk dalam hipotesis yang sama tetapi berbeda dalam faktor kepastian, maka perhitungan faktor kepastian dari kaidah yang sama dihitung dari penggabungan fungsi untuk faktor kepastian yang didefinisikan sebagai berikut:

$$CF_{combine}(CF_1, CF_2) = \begin{cases} CF_1 + CF_2(1 - CF_1) & \text{kedua} - \text{duanya} > 0 \\ \frac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)} & \text{salah satu} < 0 \\ CF_1 + CF_2(1 - CF_1) & \text{kedua} - \text{duanya} < 0 \end{cases}$$

Di mana, $CF_{combine}$ digunakan bergantung pada apakah faktor kepastian positif atau negatif.

Evidensi tidak pasti diperoleh dengan menggali dari hasil wawancara dengan pakar. Nilai $CF(Rule)$ didapat dari interpretasi term dari pakar menjadi nilai MD/MB tertentu. Suatu sistem yang menerapkan *inexact reasoning*, pertama-tama harus menemukan cara untuk mempresentasikan *uncertain evidence*. Pendekatan di atas mengubah bentuk formal dari suatu peluang $P(E)$ dengan $CF(E)$ yang berupa nilai MD/MB.

Faktor kepastian bukanlah suatu peluang, tetapi merupakan ukuran tingkat kepercayaan terhadap suatu evidensi. CF mempresentasikan tingkat kepercayaan bahwa suatu evidensi adalah benar (Bain Khusnul, 2010 : 13).

II.2.2. Evidensi Tidak Pasti (*uncertain evidence*)

Evidensi tidak pasti diperoleh dengan menggali dari hasil wawancara dengan pakar. Nilai $CF(Rule)$ didapat dari interpretasi term dari pakar menjadi nilai MD/MB tertentu. Suatu sistem yang menerapkan *inexact reasoning*, pertama-tama harus menemukan cara untuk mempresentasikan *uncertain evidence*. Pendekatan di atas mengubah bentuk formal dari suatu peluang $P(E)$ dengan $CF(E)$ yang berupa nilai MD/MB. Faktor kepastian bukanlah suatu peluang, tetapi merupakan ukuran tingkat kepercayaan terhadap suatu evidensi. CF mempresentasikan tingkat kepercayaan bahwa suatu evidensi adalah benar (Bain Khusnul, 2010 : 13).

II.3. Pengertian Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai jenis komputer dan berbagai sistem operasi termasuk telepon genggam. Java dikembangkan oleh *Sun Microsystem* dan dirilis tahun 1995. Java merupakan

suatu teknologi perangkat lunak yang digolongkan *multi platform*. Selain itu, Java juga merupakan suatu *platform* yang memiliki *virtual machine* dan *library* yang diperlukan untuk menulis dan menjalankan suatu program.

Bahasa pemrograman java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan *versi* yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, serta Sembilan pemrograman lainnya dari *Sun Microsystem*. Salah satu hasil proyek ini adalah mascot Duke yang dibuat oleh Joe Palrang (Wahana Komputer, 2010 : 1).

II.4. Pengertian Database

Database adalah koleksi data item yang saling terkait terkelola sebagai satu unit. Beberapa definisi lain tentang database adalah :

1. *Relational data structure used to store, query, and retrieve information.*
2. *A shared collection of logically related data designed to meet the information needs of multiple users in an organization.*
3. *an organized body of related information wordnet.princeton.edu/perl/webwn*
4. *database is an organized collection of computer records. In libraries, one of the most common types of databases consists of records describing articles in periodicals otherwise known as a periodical index.*

Istilah “*database*” juga digunakan di dunia perpustakaan untuk menjelaskan sekumpulan informasi yang biasanya tersimpan dan dapat diakses di perpustakaan, seperti yang dijelaskan pada definisi ke 4. Untuk membuat definisi

database yang kita maksud lebih jelas dan tepat, maka beberapa karakteristik dari database adalah :

1. pengelolaan melalui dengan suatu Database Management System (DBMS)
2. ada lapisan-lapisan abstraksi data
3. data fisik yang independen
4. data logis yang independen (Umi Proboyekti ; 2012 : 1).

II.5. Pengertian MySQL

Mysql pertama kali dirintis oleh seorang programmer database bernama Michael Widenius, yang dapat anda hubungi di emailnya monty@analytikerna. Mysql *database server* adalah RDBMS (*Relasional Database Management System*) yang dapat menangani data yang bervolume besar. meskipun begitu, tidak menuntut *resource* yang besar. Mysql adalah *database* yang paling populer di antara *database* yang lain.

MySQL adalah program *database* yang mampu mengirim dan menerima data dengan sangat cepat dan *multi user*. MySQL memiliki dua bentuk lisensi, yaitu *free software* dan *shareware*. penulis sendiri dalam menjelaskan buku ini menggunakan *database* ini untuk keperluan pribadi atau usaha tanpa harus membeli atau membayar lisensi, yang berada di bawah lisensi GNU/GPL (*general public license*) (Wahana Komputer; 2010 : 5).



Gambar II.1. Tampilan MySQL
(Sumber : Wahana Komputer, 2010 : 5)

II.6. Entity Relationship Diagram (ERD)

Entity Relationship Diagram atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analisis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata, 2010 : 67).

Tabel II.1. Simbol ERD

Notasi	Keterangan
	Entitas , adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	Relasi , menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	Atribut , berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis , sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(Sumber : Janner Simarmata, 2010 : 67)

II.7. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan sebagai berikut :

1. Bentuk normal tahap pertama (1st Normal Form)

Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri.

2. Bentuk normal tahap kedua (2nd normal form)

Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

3. Bentuk normal tahap ketiga (3rd normal form)

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya.

4. Boyce Code Normal Form (BCNF)

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

5. Bentuk Normal Tahap Keempat dan Kelima

Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Janner Simarmata, 2010 : 76).

II.8. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

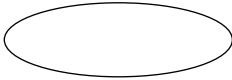
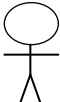
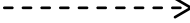
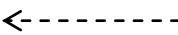
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasis UML adalah sebagai berikut :

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.2. Simbol Use Case

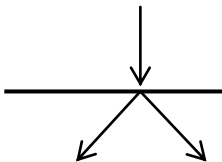
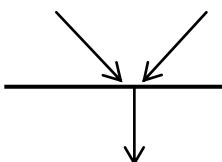
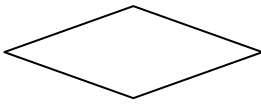
Gambar	Keterangan
	Use case menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama use case.
	Aktor adalah <i>abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan use case, tetapi tidak memiliki control terhadap use case.
	Asosiasi antara aktor dan use case, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
	Asosiasi antara aktor dan use case yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
	<i>Include</i> , merupakan di dalam use case lain (<i>required</i>) atau pemanggilan use case oleh use case lain, contohnya adalah pemanggilan sebuah fungsi program.
	<i>Extend</i> , merupakan perluasan dari use case lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata, 2013 : 4)

2. Diagram Aktivitas (Activity Diagram)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.3. Simbol Activity Diagram

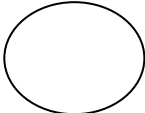
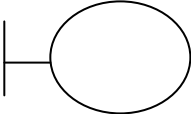
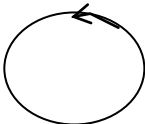
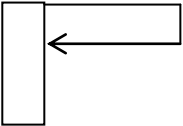
Gambar	Keterangan
	<i>Start point</i> , diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	<i>End point</i> , akhir aktifitas.
	<i>Activites</i> , menggambarkan suatu proses/kegiatan bisnis.
	<i>Fork</i> (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	<i>Join</i> (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	<i>Decision Points</i> , menggambarkan pilihan untuk pengambilan keputusan, <i>true</i> , <i>false</i> .
	<i>Swimlane</i> , pembagian <i>activity</i> diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata, 2013 : 6)

3. Diagram Urutan (Sequence Diagram)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	<i>Entity Class</i> , merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	<i>Boundary Class</i> , berisi kumpulan kelas yang menjadi <i>interface</i> atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan <i>form</i> cetak.
	<i>Control class</i> , suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	<i>Message</i> , simbol mengirim pesan antar <i>class</i> .
	<i>Recursive</i> , menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	<i>Activation</i> , <i>activation</i> mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	<i>Lifeline</i> , garis titik-titik yang terhubung dengan objek, sepanjang <i>lifeline</i> terdapat <i>activation</i> .

(Sumber : Windu Gata, 2013 : 7)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan

atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.5. Multiplicity Class Diagram

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata, 2013 : 9)