BAB II
KAJIAN PUSTAKA

2.1 Teori Dasar

2.1.1 Kecerdasan Buatan

Menurut Sutojo, Mulyanto, & Suhartono (2011: 1) kecerdasan buatan berasal dari bahasa Inggris "*Artificial Intelligence*" atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berpikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

AI (*Artificial Intelligence*) atau yang biasa disebut dengan kecerdasan buatan mempunya 3 komponen yaitu:

1. Jaringan Saraf Tiruan

   Jaringan saraf tiruan adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen kunci dari paradigma ini adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (*neuron*), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja JST seperti cara kerja manusia, yaitu belajar melalui contoh.

2. Sistem Pakar

Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna. Dengan bantuan sistem pakar seseorang yang bukan pakar/ahli dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar. Beberapa metode yang digunakan dalam sistem pakar adalah *Metode Forward Chaining*, *Metode Backward Chaining, Metode Dempster-Shafer, Metode Certainty Factor, dan Metode Probabilitas Bayesian*.

3. Logika *Fuzzy/Fuzzy Logic*

Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang cocok untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded system*, jaringan *PC*, *multi-channel* atau *workstation* berbasis akuisi data, dan sistem kontrol. Metodologi ini dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya. Beberapa metode dalam logika *fuzzy* yaitu Metode *Mamdani,* Metode *Sugeno,* Metode *Tahani*, dan Metode *Tsukamoto*.

2.1.2 Sistem Pakar

Menurut Sutojo, *et al* (2011: 159) sistem pakar merupakan cabang dari *Artificial Inteligence* (AI) yang cukup tua karena sistem ini mulai dikembangkan pada pertengahan tahun 1960. Sistem pakar yang muncul pertama kali adalah *General-purpose problem solver* (GPS) yang dikembangkan oleh Newel dan Simon. Sampai saat ini sudah banyak sistem pakar yang dibuat, seperti *MYCIN* untuk diagnosis penyakit, *DENDRAL* untuk mengidentifikasi struktur molekul campuran yang tak dikenal, *XCON* & *XSEL* untuk membantu konfigurasi sistem komputer besar, *SOPHIE* untuk analisis sirkuit elektronik, *Prospector* digunakan di bidang geologi untuk membantu mencari dan menemukan deposit, *FOLIO* digunakan unuk membantu memberikan keputusan bagi seorang manager dalam stok dan investasi, *DELTA* dipakai untuk pemeliharaan lokomotif listrik diesel, dan sebagainya.

Sedangkan menurut para ahli ada beberapa penjelasan tentang sistem pakar yaitu:

1. Martin dan Oxman (1998) dalam Hartati dan Iswanti (2008: 3) sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah, yang biasanya hanya dapat diselesaikan oleh seorang pakar dalam bidang tertentu.
2. Ignizio (1991) dalam Hartati dan Iswanti (2008: 3) sistem pakar merupakan bidang yang dicirikan oleh sistem berbasis pengetahuan (*knowledge base system*), memungkinkan komputer dapat berfikir dan mengambil kesimpulan dari sekumpulan kaidah.

3. Turban dan Aronson (2001) dalam Hartati dan Iswanti (2008: 3) sistem yang menggunakan pengetahuan manusia yang dimasukkan ke dalam komputer untuk memecahkan masalah-masalah yang biasanya diselesaikan oleh pakar.
4. Giarratano dan Riley (2005) dalam Hartati dan Iswanti (2008: 3) salah satu cabang kecerdasan buatan yang menggunakan pengetahuan-pengetahuan khusus yang dimiliki oleh seorang ahli untuk menyelesaikan suatu masalah tertentu.

2.1.2.1 Manfaat dan Kemampuan Sistem Pakar

Menurut Merlina dan Hidayat (2012: 4) manfaat dan kemampuan sistem pakar adalah sebagai berikut:

1. Meningkatkan *output* dan produktivitas.
2. Menurunkan waktu dan pengambilan keputusan.
3. Meningkatkan kualitas proses dan produk.
4. Mengurangi *downtime*.
5. Menyerap keahlian langka.
6. Fleksibilitas.
7. Operasi peralatan yang lebih mudah.
8. Eliminasi kebutuhan peralatan yang mahal.
9. Operasi di lingkungan yang berbahaya.
10. Aksesibilitas ke pengetahuan dan *help desk*.

11. Kemampuan untuk bekerja dengan informasi yang tidak lengkap/tidak pasti.
12. Kelengkapan pelatihan.
13. Peningkatan pemecahan masalah dan pengambilan keputusan.
14. Meningkatkan proses pengambilan keputusan.
15. Meningkatkan kualitas keputusan.
16. Kemampuan untuk memecahkan persoalan kompleks.
17. Transfer pengetahuan ke lokasi terpencil.

2.1.2.2 Keterbatasan Sistem Pakar

Menurut Merlina dan Hidayat (2012: 4) kelemahan sistem pakar, adalah sebagai berikut:

1. Pengetahuan tidak selalu siap tersedia.
2. Akan sulit mengekstrak keahlian dari manusia.
3. Pendekatan tiap pakar pada suatu penilaian situasi mungkin berbeda, tetapi benar.
4. Sulit, bahkan bagi pakar berkemampuan tinggi untuk mengikhtisarkan penelitian situasi yang baik pada saat berada dalam tekanan waktu.
5. Penggunaan sistem pakar memiliki batasan kognitif alami.
6. Sistem pakar bekerja dengan baik hanya dalam *domain* pengetahuan sempit.
7. Kebanyakan pakar tidak memiliki sarana mandiri untuk memeriksa apakah kesimpulannya masuk akal.
8. Kosa kata yang digunakan pakar untuk menyatakan fakta dan hubungan.

2.1.2.3 Ciri-Ciri Sistem Pakar

Ciri-ciri sistem pakar menurut Sutojo, *et al* (2012: 162) adalah sebagai berikut:

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan-alasan dengan cara yang dapat dipahami.
4. Bekerja berdasarkan kaidah/*rule* tertentu.
5. Mudah dimodifikasi.
6. Basis pengetahuan dan mekanisme *inferensi* terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pengguna.

Menurut Merlina dan Hidayat (2012: 1) jenis-jenis pengetahuan yang dimiliki dalam kepakaran adalah sebagai berikut:

1. Teori-teori dari permasalahan.
2. Aturan dan prosedur yang mengacu pada area permasalahan.
3. Aturan (*heuristic*) yang harus dikerjakan pada situasi yang terjadi.
4. Strategi global untuk menyelesaikan berbagai jenis masalah.
5. *Meta-knowledge* (pengetahuan tentang pengetahuan).
6. Fakta-fakta.

2.1.3 *Forward Chaining*

Menurut Sutojo, *et al* (2011: 171) *forward chaining* adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian *IF* dari *rules IF-THEN*. Bila ada fakta yang cocok dengan bagian *IF*, maka *rule* tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian *THEN*) di tambahkan ke dalam database. Setiap kali pencocokan, dimulai dari *rule* teratas. Setiap *rule* hanya boleh dieksekusi sekali saja. Proses pencocokan berhenti bila tidak ada lagi *rule* yang bisa dieksekusi. Metode pencarian yang digunakan adalah *Depth-First Search* (DFS), *Breath-First Search* (BFS) atau *Best-First Search.*

Menurut Merlina dan Hidayat (2012: 22) *forward chaining* adalah pendekatan *data-driven* yang dimulai dari informasi yang tersedia atau dari ide dasar, kemudian mencoba menarik kesimpulan.

Data aturan kesimpulan:

A = 1 *IF* A = 1 *AND* B =2

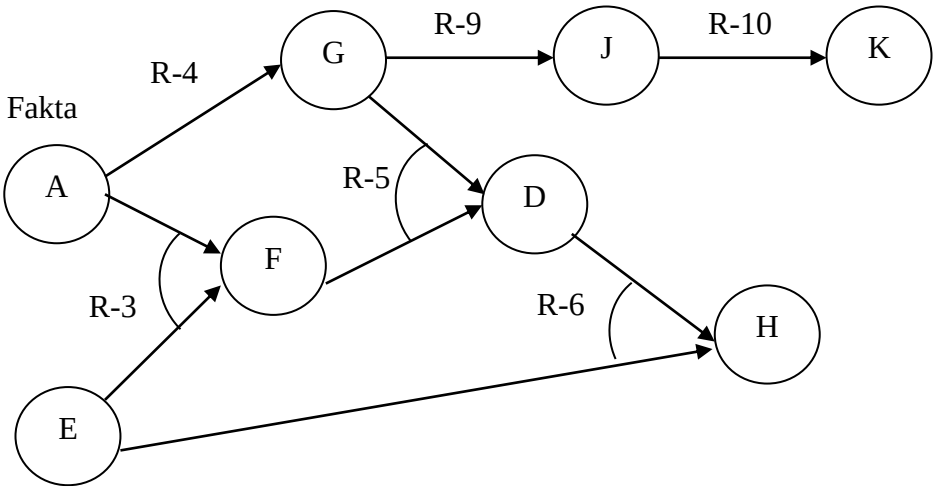
B = 2 *THEN* C = 3 C = 3

Contoh:

IF akar tanaman rusak

AND terdapat telur-telur ulat pada rerumputan

THEN terserang hama ulat grayak



Gambar 2.1 Cara Kerja Mesin Inferensi (*Forward Chaining*)
Sumber: Merlina dan Hidayat (2012)

2.1.4 Pohon Keputusan

Menurut Merlina dan Hidayat (2012: 13) setiap permasalahan yang kita hadapi memiliki perbedaan dalam hal kompleksitas dan kesulitan, mulai dari permasalahan yang sederhana sampai dengan permasalahan yang rumit harus diukur faktor-faktor yang mempengaruhi dari permasalahan tersebut, hal ini diperlukan sebuah analisis untuk memperhitungkan jalan keluar bagi permasalahannya. Untuk menyelesaikan permasalahan secara optimal.

Pohon keputusan adalah sebuah jawaban akan sebuah sistem/cara yang kita kembangkan untuk membantu mencari dan membuat keputusan untuk masalah-

masalah tersebut. Dengan pohon keputusan, manusia dapat dengan mudah mengidentifikasi dan melihat hubungan antara faktor-faktor yang mempengaruhi suatu masalah dan dapat mencari penyelesaian terbaik dengan memperhitungkan faktor-faktor tersebut.

Pohon keputusan juga dapat menganalisis nilai risiko dan nilai suatu informasi yang terdapat dalam suatu alternatif pemecahan masalah. Peranan pohon keputusan sebagai alat bantu dalam mengambil keputusan (*decision support tool*).

2.1.4.1 Tabel Keputusan

Menurut Hartati dan Iswanti (2008: 26) tabel keputusan merupakan suatu cara untuk mendokumentasikan pengetahuan. Tabel keputusan merupakan matrik kondisi yang dipertimbangkan dalam pendeskripsian kaidah.

2.1.5 Mesin Inferensi

Menurut Merlina dan Hidayat (2012: 6) mesin inferensi merupakan otak dari sistem pakar. Komponen ini sebenarnya (pertimbangan) mengenai informasi dalam basis pengetahuan dan dalam "*workplace*", dan digunakan untuk merumuskan kesimpulan. Mesin inferensi mempunyai 3 elemen utama, yaitu sebagai berikut:

1. *Interpreter*, adalah elemen yang mengeksekusi item agenda yang dipilih dengan mengaplikasikannya pada basis pengetahuan *rule* yang berhubungan.
2. *Scheduler,* adalah elemen yang menjaga kontrol di sepanjang agenda. Memperkirakan akibat dari pengaplikasian *rule inferensia* yang menampilkan priorotas item atau kriteria lain pada agenda.
3. *Consistency enforcer*, adalah elemen yang mencoba menjaga konsistensi representasi solusi yang muncul.

2.2 Variabel

Menurut Sudaryono (2015: 16) variabel penelitian pada dasarnya adalah segala sesuatu yang berbentuk apa saja yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi dan kesimpulannya. Penulis menetapkan variabel penelitian dalam penelitian ini adalah kerusakan mesin kopi venusta. Sedangkan indikator dalam penelitian ini adalah sekat geser, set kahalusan, grinder, brewer, corong canister, mixing bowl, vapor trap, selang, dudukan selang.

2.3 Software Pendukung

2.3.1 *Unified Modeling Language* (UML)

Menurut Rosa dan Salahudin (2011: 113) UML (*Unified modeling* language) adalah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Pada perkembangan perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar semua orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

2.3.1.1 *Use Case* Diagram

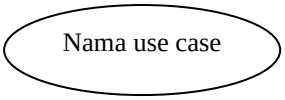
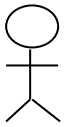
Menurut Rosa dan Salahudin (2011: 130) *use case* atau diagram *use case* merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor

dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu:

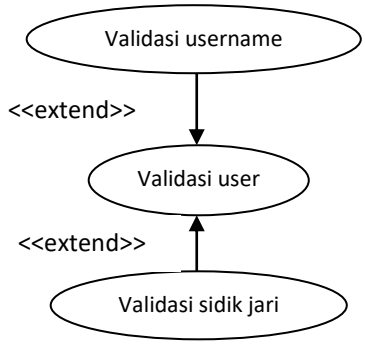
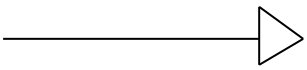
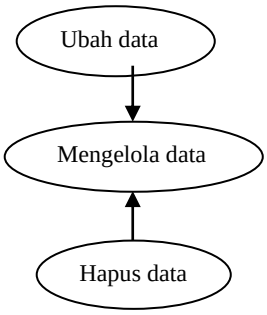
1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit saling bertukar pesan antara unit atau aktor.

Berikut ini adalah simbol-simbol yang ada pada diagram *use case*:

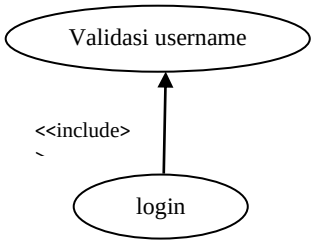
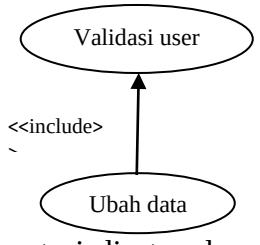
Tabel 2.1 Simbol-simbol *Use Case* Diagram

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frasa nama *use case.*
Aktor / *actor*  Nama *actor*	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat iu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi / *association* ———	Komunikasi antara aktor dan *use* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi / *extend*	Relasi *use case* tambahan ke *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use.*

Tabel 2.1 Lanjutan

<<extend>>	*Case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan.
Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)

Tabel 2.1 Lanjutan

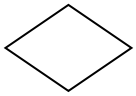
Menggunakan / *include* / *uses*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan
<<include>> ……..> <<users>> ——▷	Fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang yang cukup besar mengenai include di *use case*: 1. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, missal pada kasus berikut:  2. *Include* berarti *use case* yang ditambahkan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, missal pada kasus berikut:  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan

Sumber : Rosa dan Salahudin (2011)

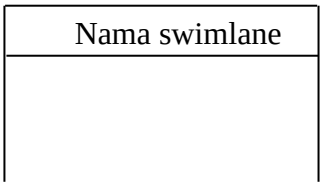
2.3.1.2 *Activity* Diagram

Menurut Rosa dan Salahudin (2011: 134) diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2.2 Simbol-simbol *Activity* Diagram

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join* 	Asosiasi penggabungan simana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah satus akhir

Tabel 2.2 Lanjutan

Swimlane  Atau  	Memisahkan organisasi bisnis bertanggung jawab terhadap aktivitas yang terjadi

Sumber: Rosa dan Salahudin (2011)

2.3.1.3 *Sequence* Diagram

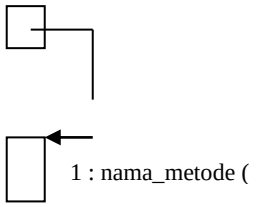
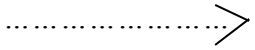
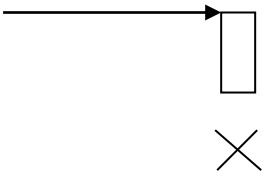
Menurut Rosa dan Salahudin (2011: 137) diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antara objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Tabel 2.3 Simbol-simbol pada *sequence* diagram

Simbol	Deskripsi
Aktor  nama aktor atau Nama aktor tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frasa nama aktor
Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek
Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif berinteraksi pesan
Pesan tipe *create*	Menyatakan suatu objek membuat

Tabel 2.3 Lanjutan

<<create>> 	Objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe *call* 1 : nama_metode( ) 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode makan operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe *send* 1 : masukan 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe *return* 1: keluaran 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* <<destroy>> 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

Sumber: Rossa dan Salahudin (2011)

2.3.1.4 *Class* Diagram

Menurut Rosa dan Salahudin (2011: 122) diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi:

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai denan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas sebagai berikut:

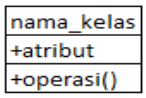
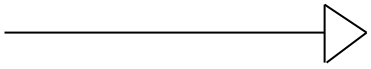
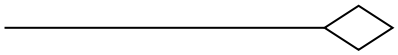
1. Kelas *main*, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
4. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas

utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.4 Simbol-simbol pada *class* diagram

Simbol	**Deskripsi**
Kelas 	Kelas pada struktur sistem
Antarmuka / *interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *assosiation*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *directed association*	Relasi antar kelas yang makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multipicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
Agregasi / *aggregation*	Relasi antar kelas dengan makna
	Semua-bagian (*whole-part*)

Sumber: Rosa dan Salahudin (2011)

2.3.2 Bahasa Pemrograman *PHP*

Menurut Raharjo, *et al* (2010: 41) *PHP* adalah salah satu bahasa pemrograman skrip yang dirancang untuk membangun aplikasi *web*. Ketika dipanggil dari *web browser*, program yang ditulis dengan *php* akan di*parsing* di dalam *web server* oleh *interpreter php* dan diterjemahkan ke dalam dokumen *html*, yang selanjutnya akan ditampilkan kembali ke *web server*, *php* dikatakan sebagai bahasa sisi *server* (*server-side*). Oleh sebab itu, seperti yang telah dikemukakan sebelumnya, kode *php* tidak akan terlihat pada saat user memilih perintah "*view source*" pada *web browser* yang mereka gunakan. Selain menggunakan *php*, aplikasi web juga dapat dibangun dengan *java* (*JSP - Java Server Pages dan Servlet*), *Perl*, maupun *ASP* (*Active Server Pages*).

2.3.3 *PHP MyAdmin*

Menurut Kadir (2009: 30) *PhpMyAdmin* adalah utilitas yang tersedia pada *WAMP5*, yang dapat digunakan untuk berinteraksi dengan *database mysql*. Utilitas ini berbasis *web* dan dapat digunakan untuk melakukan berbagai operasi yang mengakses *database*.

2.3.4 *XAMPP*

Menurut Aditya (2011: 16) *xampp* adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program.

Fungsinya adalah sebagai *server* yang berdiri sendiri (*localhost*), yang terdiri atas program *Apache HTTP Server, My SQL database,* dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama *xampp* merupakan singkatan dari X (empat sistem operasi apapun), *Apache, My SQL, PHP* dan *Perl*. Program ini tersedia dalam GNU (*General Public License)* dan bebas, merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis.

2.3.5 *Framework Bootstrap*

Menurut Mulhim (2014: 150) *bootstrap* merupakan sebuah *framework* yang dikembangkan oleh tim *twitter* yang berfungsi sebagai kerangka kerja untuk memudahkan dan mempercepat dalam membuat tampilan *website* yang *responsif* dan terstandar. *Bootstrap* menyediakan komponen-komponen antarmuka siap pakai yang telah dirancang sedemikian rupa untuk digunakan dalam membuat tampilan halaman *website responsif*. Selain komponen-komponen dasar, *bootstrap* juga menyediakan sarana untuk membuat *layout* halaman *web* dengan mudah dan rapi yaitu menggunakan *grid system* 12 kolom.

Bootstrap dibangun dengan teknologi *HTML, CSS,* dan *Javascript* yang dapat membuat *layout* halaman *website*, tabel, tombol, *form*, navigasi, dan komponen lainnya dalam sebuah *website* hanya dengan memanggil fungsi *class* dalam berkas *HTML* yang telah didefinisikan.

2.4 Penelitian Terdahulu

Berikut ini ada beberapa penelitian terdahulu yang diambil dari beberapa jurnal ilmiah, sebagai bahan penelitian dan referensi dalam penelitian ini.

1. Honggowibowo, Anton Setiawan (2009: 187) dengan judul "Sistem Pakar Diagnosa Penyakit Tanaman Padi Berbasis *Web* Dengan *Forward Chaining* dan *Backward Chaining*", ISSN: 1693-6930. Hasil dari implementasi: sistem pakar diagnosa penyakit tanaman padi dengan metode *inferensi forward chaining* dan *backward chaining* berbasis *web* mempermudah untuk diakses oleh siapa saja (khususnya petani) dan dimana saja (asalkan tersedia jaringan internet).
2. Wahyudi dan Utami (2011: 2) dengan judul "Sistem Pakar Diagnosa Penyakit Pada Ayam Dengan Metode *Forward Chaining*", ISSN: 1858 - 2680. Metode inferensi yang digunakan adalah *forward chaining*, yaitu: proses inferensi yang memulai pencarian dari premis atau data masukan berupa gejala menuju pada konklusi yaitu kesimpulan penyakit yang diderita serta memberikan solusi mengenai saran pengobatan dan pencegahan berdasarkan gejala-gejala yang diamati. Dari pengujian sistem yang dilakukan, sistem pakar ini telah sesuai dengan pengetahuan dan keahlian seorang pakar.
3. Benny Wijaya & Maria Prasetiyowati (2012: 1) dengan judul "Rancang Bangun Sistem Pakar Pendiagnosa Penyakit Demam *Typhoid* dan Demam Berdarah *Dengue* dengan Metode *Forward Chaining*", ISSN: 2085-4552. Sistem pakar ini dirancang dan dibangun agar dapat membantu orang-orang

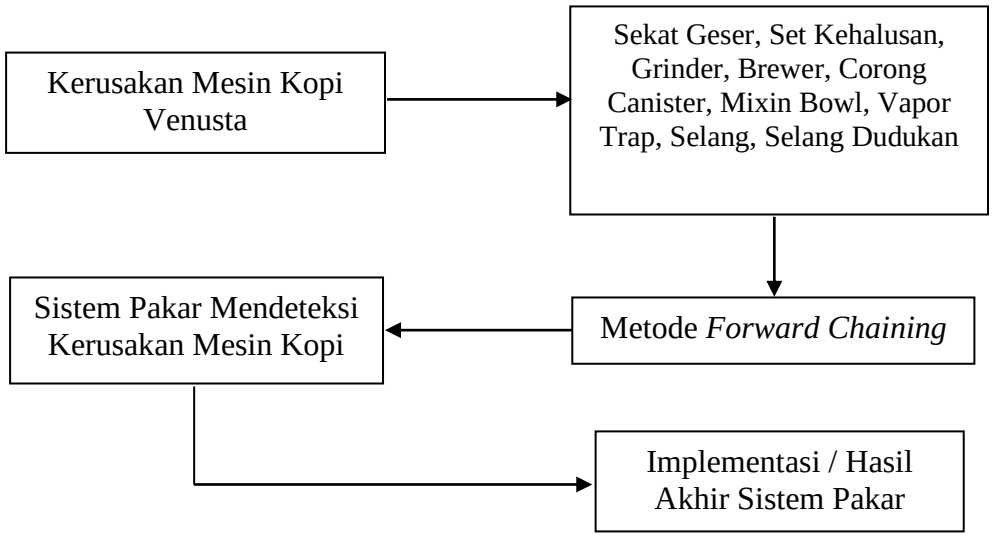
khususnya orang-orang yang bekerja di dunia kesehatan untuk melakukan diagnosa penyakit demam *typhoid* dan demam berdarah *dengue* dengan benar. Hasil dari implementasi: tingkat keakuratan sistem bergantung pada *knowledge base* yang disimpan dalam database.

4. Setyowati dan Permana (2013: 1) dengan judul "Sistem Pakar Diagnosa Penyakit Tanaman Padi Berbasis *Web*", ISSN: 1979-8415. Perancangan *database* bertujuan untuk membentuk struktur tabel yang saling terkait sehingga membentuk suatu *database* relasional. Perancangan ini bertujuan untuk menentukan atribut, *type* data beserta ukuran pada masing-masing tabel. Berdasarkan proses *desain system* dan tahapan implementasi diperoleh hasil berupa sistem pakar berkaitan dengan penyakit pada tanaman padi. Hasil *desain system* telah mampu melakukan penelusuran terhadap penyakit tanaman padi.
5. Harison dan Alexyusanderia (2014: 2) dengan judul "Sistem Pakar Perawatan Dan Perbaikan Ringan Mobil Bensin Menggunakan Video Tutorial Berbasis *Web*", ISSN: 1693-752X. Data utama di dapat melalui wawancara langsung dengan pakar yang dijadikan sampel penelitian. Kemudian untuk mendukung data yang sudah ada, penulis juga melakukan turun langsung ke bengkel untuk mendapatkan data tambahan sesuai dengan yang penulis butuhkan. Tahapan proses dalam penelitian ini mengalir sesuai dengan alur yang logis. Tujuannya adalah memberikan petunjuk yang jelas, teratur dan sistematis. Hasil implementasi/kesimpulannya: sistem pakar perawatan dan perbaikan ringan mobil bensin menggunakan video tutorial

berbasis web dibangun memberikan kemudahan dalam memberikan petunjuk dalam menentukan bagian komponen mobil yang mengalami kerusakan karna setiap kendala kerusakan yang didiagnosis sistem ini memberikan video komponen mobil tersebut.

2.5 Kerangka Pemikiran

Kerangka penelitian dalam penelitian ini adalah sebagai berikut:



Gambar 2.2 Kerangka Pemikiran
Sumber: Olahan Data Peneliti (2017)