# BAB II

# LANDASAN TEORI

Pembelajaran berbasis komputer digunakan sejak tahun 1960 di Amerika Serikat, pada awalnya karena keterbatasan teknologi bentuk pembelajaran berbasis komputer masih sangat sederhana dan kurang berhasil menarik minat pendidik untuk memanfaatkannya. Pada perkembangan selanjutnya banyak aplikasi pembelajaran berbasis komputer yang disempurnakan dan dianggap baik untuk diterapkan disekolah-sekolah karena kemajuan perangkat lunak dan teknologi komputer terus berkembang.

Antara tahun 1982 dan 1983 jumlah anak SD yang menggunakan computer meningkat dari 20% menjadi 62%. Sekolah Menengah Pertama dan Sekolah Lanjutan Atas, kenaikannya adalah 40% ke 81% dan dari 58% ke 86%. Peningkatan tersebut mengejutkan para ahli bidang pendidikan. Diperkirakan pada tahun 1990 ada 9 juta komputer yang akan digunakan disekolah-sekolah Amerika Serikat. Perusahaan perangkat lunak berusaha mengambil manfaat dari peningkatan pengguna komputer tersebut seperti: science Researce Assosiates dan Central Data Corporation. Dua sistem pembelajaran berbasis komputer yang terkenal di Amerika Serikat pada pertengahan dasawarsa delapan puluh tahun adalah Plato dan Ticcit.

Penggunaan komputer di Amerika Serikat tersebut diikuti oleh negara-negara lainnya yang memang memiliki teknologi komputer sejajar dengan Amerika Serikat seperti Inggris dan Jepang. Mereka berusaha menerapkan penggunaan pembelajaran berbasis komputer dilokasi yang dianggap membutuhkan pembelajaran berbasis

komputer dalam menunjang porses belajar mengajar. Pada perkembangan selanjutnya pembelajaran berbasis komputer dalam dunia pendidikan mulai meluas ke negara berkembang seperti Indonesia.

Penggunaan metode pembelajaran berbasis komputer diharapkan mampu membantu fungsi guru dalam menyampaikan suatu kategori kepada pihak yang mempelajari kategori tersebut, sehingga siapa saja dapat belajar individu.

Sistem pengajaran individu dipandang perlu karena adanya perbedaan pada orang yang belajar. Perbedaan yang ada antara lain kebutuhan masing-masing dalam mempelajari suatu kategori, karakteristik belajar atau cara belajar dan tingkat pemahaman terhadap suatu kategori.

## 2.1 Kemampuan Pengajaran Berbantuan Komputer

a. Perangkat Keras

Perangkat Keras merupakan perangkat fisik yang berhubungan dengan sistem pembelajaran berbasis komputer, termasuk didalamnya komputer (CPU) dan perlengkapannya seperti : disk drive, printer, monitor, papan kunci, piranti multimedia.

b. Perangkat Lunak

Perangkat Lunak dalam hubungannya dengan pembelajaran berbasis komputer adalah semua program yang mendukung pelaksanaan pembelajaran berbasis komputer, mulai dari perancangan sampai implementasi. Perangkat lunak yang dimaksud meliputi sistem oprerasi, perangkat lunak untuk pengembangan

modul, perangkat lunak aplikasi seperti mengelola data. Modul pembelajaran berbasis komputer merupakan program komputer yang digunakan untuk merepresentasikan kategori pelajaran.

Karakteristik modul pembelajaran berbasis komputer meliputi :

1. Adanya strategi penyampaian instruksional, seperti menunjukan pada kurikulum pendidikan yang ada. Khusus modul pembelajaran berbasis komputer yang dirancang untuk pelatihan, modul itu harus dikembangkan sesuai dengan pesanan, berdasarkan analisa tugas dan analisa pekerjaan.

2. Kemudian dikirimkan karakteristik ini menyangkut mobilitas modul. Bagaimana agar modul ini dapat digunakan secara bersama-sama dalam lokasi yang berbeda. Mobilitas modul bias dilakukan baik secara fisik (pengiriman disket) dan elektronik (dengan jaringan komputer ).

3. Perangkat Manusia Semua orang dengan keahlian masing-masing yang dibutuhkan untuk mengembangkan, mengoperasikan, memelihara dan mengevaluasi pembelajaran berbasis komputer. Dalam pemilihan dan pengimplementasian sistem pembelajaran berbasis komputer, ketiga komponen diatas memiliki pertimbangan yang sama, tanpa dukungan perangkat yang berkualitas tinggi pun tidak bisa dioptimalkan. Hal yang sama akan terjadi jika terdapat perangkat lunak yang baik atau tidak adanya manusia yang professional dan pemakai yang sesuai dalam pelaksanaan pembelajaran berbasis komputer.

## 2.2 Konsep Dasar Pendidikan Bahasa Inggris

Berikut ini adalah pengertian konsep dasar pendidikan bahasa Inggris

### 2.2.1. Pengertian Pendidikan

Pendidikan adalah usaha sadar dan terencana untuk mewujudkan suasana belajar dan proses pembelajaran agar peserta didik secara aktif mengembangkan potensi dirinya untuk memiliki kekuatan spiritual keagamaan, pengendalian diri, kepribadian, kecerdasan, akhlak mulia, serta keterampilan yang diperlukan dirinya, masyarakat, bangsa, dan negara.

#### 2.2.1.1.Pedidikan Nasional

Pendidikan nasional adalah pendidikan yang berdasarkan Pancasila dan Undang- Undang Dasar Negara Republik Indonesia Tahun 1945 yang berakar pada nilai-nilai agama, kebudayaan nasional Indonesia dan tanggap terhadap tuntutan perubahan zaman. Sistem pendidikan nasional adalah keseluruhan komponen pendidikan yang saling terkait secara terpadu untuk mencapai tujuan pendidikan nasional.

#### 2.2.1.2.Jalur Pendidikan

Jalur pendidikan adalah wahana yang dilalui peserta didik untuk mengembangkan potensi diri dalam suatu proses pendidikan yang sesuai dengan tujuan pendidikan.

### 2.2.1.3.Jenjang pendidikan

Jenjang pendidikan adalah tahapan pendidikan yang ditetapkan berdasarkan tingkat perkembangan peserta didik, tujuan yang akan dicapai, dan kemampuan yang dikembangkan.

### 2.2.1.4.Satuan pendidikan

Satuan pendidikan adalah kelompok layanan pendidikan yang menyelenggarakan pendidikan pada jalur formal, nonformal, dan informal pada setiap jenjang dan jenis pendidikan.

a. Pendidikan formal

Pendidikan formal adalah jalur pendidikan yang terstruktur dan berjenjang yang terdiri atas pendidikan dasar, pendidikan menengah, dan pendidikan tinggi.

b. Pendidikan nonformal

Pendidikan nonformal adalah jalur pendidikan di luar pendidikan formal yang dapat dilaksanakan secara terstruktur dan berjenjang.

c. Pendidikan informal

Pendidikan informal adalah jalur pendidikan keluarga dan lingkungan.

### 2.2.2. Pengertian Bahasa

Menurut Keraf dalam Smarapradipha (2005:1), memberikan dua pengertian bahasa. Pengertian pertama menyatakan bahasa sebagai alat komunikasi antara anggota masyarakat berupa simbol bunyi yang dihasilkan oleh alat ucap manusia. Kedua, bahasa adalah sistem komunikasi yang mempergunakan simbol-simbol vokal (bunyi ujaran) yang bersifat arbitrer.
Menurut Wibowo (2001:3), bahasa adalah sistem simbol bunyi yang bermakna dan berartikulasi (dihasilkan oleh alat ucap) yang bersifat arbitrer dan konvensional, yang dipakai sebagai alat berkomunikasi oleh sekelompok manusia untuk melahirkan perasaan dan pikiran.

### 2.2.3. Pengertian Bahasa Inggris

Bahasa Inggris adalah media komunikasi utama bagi masyarakat di negara Inggris, Amerika Serikat, Kanada, Australia, New Zealand, Afrika Selatan, dan di banyak negara lainnya. Bahasa Inggris merupakan bahasa resmi dari banyak negara-negara persemakmuran dan dipahami serta dipergunakan secara meluas. Bahasa Inggris dipergunakan di lebih banyak negara di dunia dibanding bahasa yang lain serta dibanding bahasa yang lain kecuali bahasa Cina, bahasa ini juga lebih banyak dipergunakan orang. Bahasa Inggris termasuk rumpun bahasa-bahasa Anglo-Frisia pada cabang barat bahasa-bahasa Jerman, dan merupakan sebuah bahasa subfamili dari bahasa-bahsa Indo-Eropa. Bahasa Inggris hampir mendekati bahasa Frisia, sedikit lebih luas dari bahasa Netherlandic (Belanda –Flemish) dan dialek Jerman tingkat rendah (Plattdeutsch), serta jauh dari bahasa Jerman Modern tingkat tinggi.

## 2.3 Konsep Dasar Perangkat Lunak

Konsep dasar perangkat lunak dapat dilihat dari pengertian perangkat lunak, fungsi perangkat lunak, dan macam perangkat lunak.

### 2.3.1. Pengertian Perangkat Lunak

Dalam arti sempit perangkat lunak adalah program yang di jalankan di suatu pemroses. Saat ini, hampir seluruh bidang kehidupan manusia telah melibatkan perangkat lunak. Hamper seluruh perangkat yang digunakan manusia telah dilengkapi pemroses. Perangkat seperti handphone, televise, mesin cuci, mesin AC, mobil mewah dan sebagainya telah menjadi cerdas dikarenakan program didalamnya.

Manusia telah mempersiapkan diri untuk perluasan penggunaan perangkat lunak di semua perangkat yang di pakai. Dengan lebar alamat 128 bit, maka cakupan IPng ( IP versi 6-*next generation* ) dirancang mencakup hamper seluruh titik di muka bumi ini. Cakupan itu berarti di titik-titik itu akan di pasha perangkat yang didalamnya berisi perangkat lunak (program) yang berkomunikasi menggunakan protocol TCP/IP.

Perangkat lunak adalah produk yang seharusnya dirancang dan dibangun lewat aktivitas-aktivitas rekayasa perangkat lunak yang berdisiplin dan sistematis. Perangkat lunak dalam arti lebih luas terdiri dari program-program yang di eksekusi computer dalam beraneka ukuran dan beraneka ragam arsitektur, dokumen-dokumen berupa *hard-copy* dan bentuk-bentuk maya, dan data berupa angka-angka dan teks juga representasi informasi gambar, video dan audio.

Perangkat lunak juga dapat di artikan juga program yang berisi kumpulan instruksi untuk melakukan proses pengolahan data. *Software* sebagai penghubung antara manusia sebagai pengguna dengan perangkat keras komputer, berfungsi menerjemahkan bahasa manusia ke dalam bahasa mesin sehingga perangkat keras komputer memahami keinginan pengguna dan menjalankan instruksi yang diberikan dan selanjutnya memberikan hasil yang diinginkan oleh manusia tersebut.

### 2.3.2. Fungsi Perangkat lunak

Perangkat lunak komputer berfungsi untuk :

a. Mengidentifikasi program

b. Menyiapkan aplikasi program sehingga tata kerja seluruh perangkat komputer terkontrol.
c. Mengatur dan membuat pekerjaan lebih efisien.

### 2.3.3. Macam-macam Perangkat Lunak

Perangkat lunak terbagi menjadi 4 macam, yaitu :

a. Sistem Operasi (*Operating System*),
b. Program Aplikasi (*Application Programs*),
c. Bahasa Pemrograman (*Programming Language*),
d. Program Bantu (*Utility*).

#### 2.3.3.1. Sistem Operasi (*Operating System*)

Sistem Operasi yaitu program yang berfungsi untuk mengendalikan sistem kerja yang mendasar sehingga mengatur kerja media input, output, tabel pengkodean, memori, penjadwalan prosesor, dan lain-lain. Sistem operasi berfungsi sebagai penghubung antara manusia dengan perangkat keras dan perangkat lunak yang akan digunakan. Adapun fungsi utama sistem operasi adalah :

a. Menyimpan program dan aksesnya.
b. Membagi tugas di dalam CPU.
c. Mengalokasikan tugas-tugas penting.
d. Merekam sumber-sumber data.
e. Mengatur memori sistem termasuk penyimpanan, menghapus dan mendapatkan data.

f. Memeriksa kesalahan sistem.

g. Multitugas pada OS/2″, Windows ’95″, Windows ’98″, Windows NT”, /2000/XP.

h. Memelihara keamanan sistem, khusus pada jaringan yang membutuhkan kata sandi (*password*) dan penggunaan ID.

Contoh Sistem Operasi, misalnya : Disk operating System (DOS), Microsoft Windows, Linux, dan Unix.

#### 2.3.3.2. Program Aplikasi (*Aplication Programs*)

Program Aplikasi adalah perangkat lunak yang dirancang khusus untuk kebutuhan tertentu, misalnya program pengolah kata, mengelola lembar kerja, program presentasi, design grafis, dan lain-lain.

#### 2.3.3.3. Bahasa Pemrograman (*Programming Language*)

Perangkat lunak bahasa yaitu program yang digunakan untuk menerjemahkan instruksi-instruksi yang ditulis dalam bahasa pemrograman ke bahasa mesin dengan aturan atau prosedur tertentu, agar diterima oleh komputer.

Ada 3 level bahasa pemrograman, yaitu :

1. Bahasa tingkat rendah (*low level language*)

Bahasa ini disebut juga bahasa mesin (assembler), dimana pengkodean bahasanya menggunakan kode angka 0 dan 1.

2. Bahasa tingkat tinggi (*high level language*)

Bahasa ini termasuk dalam bahasa pemrograman yang mudah dipelajari oleh pengguna komputer karena menggunakan bahasa Inggris. Contohnya : BASIC, COBOL, PASCAL, FORTRAN.

3. Bahasa generasi keempat (4 GL)

Bahasa pemrograman 4 GL (Fourth Generation Language) merupakan bahasa yang berorientasi pada objek yang disebut Object Oriented Programming (OOP). Contoh software ini adalah : Visual Basic, Delphi, Visual C++.

#### 2.3.3.4. Program Bantu (*Utility*)

Perangkat Lunak merupakan perangkat lunak yang berfungsi sebagai aplikasi pembantu dalam kegiatan yang ada hubungannya dengan komputer, misalnya memformat disket, mengopi data, mengkompres file, dan lain-lain.

Contoh software ini diantaranya :

a. Norton Utility
b. Winzip
c. Norton Ghost
d. Antivirus

### 2.4 Konsep *Object Oriented Programming*

Dalam pemrogramman berbasis objek, sebuah program dibagi menjadi bagian-bagian kecil yang disebut Objek. Objek tersebut dapat diolah sendiri-sendiri. Setiap objek memiliki sekumpulan sifat dan metoda yang melakukan fungsi tertentu sesuai yang telah diprogramkan kepadanya.

Pada prinsipnya sebuah objek mengandung tiga hal utama :

1. *Properti* atau *Atribut*

*Property* adalah karakteristik atau sifat dari sebuah objek. Misalnya, *property* warna untuk teks adalah merah, ukuran 10, font-nya Arial, dan sebagainya,

2. Metode (*Methode*)

Metode adalah prosedur yang dimiliki oleh suatu objek yang akan dijalankan sesuai dengan respon yang diberikan oleh suatu perintah atau kejadian. Misalkan, objek tombol *EXIT* memiliki metode keluar dari aplikasi.

*3. Event*

*Event* adalah "kejadian", atau kejadian sesuatu yang dapat dialami oleh sebuah objek. Sebagai contoh, meletakan *mouse* di atas objek, mengklik tombol *mouse* pada sebuah tombol, mengetik pada kotak teks, dan sebagainya.

Analisis dan disain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas.

Pengertian **"berorientasi objek"** berarti bahwa kita mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya.

Metode pengembangan sistem yang digunakan dalam penelitian ini yaitu dengan pengembangan berorientasi objek dimana :

a. Abstraksi yang lebih tinggi karena pengembangan dapat berlangsung di level objek dan mengabaikan rincian-rincian (yaitu data dan fungsi perlu ada) system sepanjang masih perlu. Dengan demikian menjadikan proses perencangan, pengkodean, pengujian dan pemeliharaan system jauh lebih sederhana.
b. Transisi yang *seamlessness* di antara tahap-tahap di proses pengembangan perangkat lunak. Pada hamper seluruh tahap proses pengembangan berorientasi objek maka konsep, bentukan yang sama yang digunakan.
c. Dapat memanfaatkan keampuhan bahasa berorientasi objek dan mempraktekan teknik pemrograman yang baik.
d. Objek dapat digunaulang karena memodelkan domain persoalan di dunia eksternal secara langsung setidaknya sampai model analisi. Masing-masing objek dapat berdiri sendiri atau di dalam lingkaran sempit kelas-kelas kolaboratornya. Dengan kemampuan lebih mandiri, maka tingkat gunaulang system berorientasi objek menjadi lebih tinggi di banding pendekatan konvensional.

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama :

A. *Encapsulation*

1. *Encapsulation* merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses.
2. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya.

3. Data terlindung dari prosedur atau objek lain, kecuali prosedur yang berada dalam objek itu sendiri.

B. *Inheritance*

1. *Inheritance* adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek dari objek induk diturunkan kepada anak objek, demikian seterusnya.
2. *Inheritance* mempunyai arti bahwa atribut dan operasi yang dimiliki bersama di anatara kelas yang mempunyai hubungan secara hirarki.
3. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimiliki oleh kelas induknya, dan ditambah dengan sifat unik yang dimilikinya.
4. Kelas Objek dapat didefinisikan atribut dan *service* dari kelas Objek lainnya.
5. Inheritance menggambarkan generalisasi sebuah kelas

C. *Polymorphism*

1. *Polimorfisme* yaitu konsep yang menyatakan bahwa seuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.
2. *Polimorfisme* mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.
3. Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon *message* yang sama.

4. Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan Objek.

## 2.5 Perangkat Lunak yang Digunakan

Perangkat lunak harus dimiliki oleh suatu sistem agar dapat mencapai suatu tujuan. Berikut perangkat lunak pendukung dalam aplikasi ini :

### 2.5.1. JAVA

Java adalah sebuah bahasa pemrograman komputer berbasiskan kepada *Object Oriented Programming*. Java diciptakan setelah C++ dan didesain sedemikian sehingga ukurannya kecil, sederhana, dan *portable* (dapat dipindah-pindahkan di antara bermacam platform dan sistem operasi).

#### 2.5.1.1. Pengertian Java

Java adalah sebuah teknologi yang diperkenalkan oleh *Sun Microsystems* pada pertengahan tahun 1990. Menurut definisi dari Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Kita lebih menyukai menyebut Java sebagai sebuah teknologi dibanding hanya sebuah bahasa pemrograman, karena Java lebih lengkap dibanding sebuah bahasa pemrograman konvensional. Teknologi Java memiliki tiga komponen penting, yaitu:

a. *Programming-language specification*
b. *Application-programming interface*
c. *Virtual-machine specification*

#### 2.5.1.2. Java API

Java API terdiri dari tiga bagian utama:

a. Java Standard Edition (SE), sebuah standar API untuk merancang aplikasi desktop dan *applets* dengan bahasa dasar yang mendukung grafis, M/K, keamanan, konektivitas basis data dan jaringan.

b. Java Enterprose Edition (EE), sebuah inisiatif API untuk merancang aplikasi server dengan mendukung untuk basis data.

c. Java Macro Edition (ME), sebuah API untuk merancang aplikasi yang jalan pada alat kecil seperti telepon genggam, komputer genggam dan pager.

#### 2.5.1.3. Java Virtual Machine

Java Virtual Machine (JVM) adalah sebuah spesifikasi untuk sebuah komputer abstrak. JVM terdiri dari sebuah kelas pemanggil dan sebuah interpreter Java yang mengeksekusi kode arsitektur netral. Kelas pemanggil memanggil file .class dari kedua program Java dan Java API untuk dieksekusi oleh interpreter Java. Interpreter Java mungkin sebuah perangkat lunak interpreter yang menterjemahkan satu kode byte pada satu waktu, atau mungkin sebuah just-intime (JIT) kompiler yang menurunkan *bytecode* arsitektur netral kedalam bahasa mesin untuk *host computer*.

#### 2.5.1.4. Sistem Operasi Java

Sistem operasi biasanya ditulis dalam sebuah kombinasi dari kode bahssa C dan assembly, terutama disebabkan oleh kelebihan performa dari bahasa tersebut dan memudahkan komunikasi dengan perangkat keras.

Satu kesulitan dalam merancang sistem basis bahasa adalah dalam hal proteksi memori, yaitu memproteksi sistem opeasi dari pemakai program yang sengaja memproteksi pemakai program lainnya. Sistem operasi tradisional mengaharapkan pada tampilan perangkat keras untuk menyediakan proteksi memori. Sistem basis bahasa mengandalkan pada tampilan keamanan dari bahasa. Sebagai hasilnya, sistem basis bahasa menginginkan pada alat perangkat keras kecil, yang mungkin kekurangan tampilan perangkat keras yang menyediakan proteksi memori.

#### 2.5.1.5. Dasar Pemrograman

Java2 adalah generasi kedua dari Java *platform* (generasi awalnya adalah Java Development Kit). Java berdiri di atas sebuah mesin interpreter yang diberi nama JVM. JVM inilah yang akan membaca *bytecode* dalam file *.class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu, bahasa Java disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM.

*Platform* Java terdiri dari kumpulan *library*, JVM, kelas- kelas *loader* yang dipaket dalam sebuah lingkungan rutin Java, dan sebuah *compiler, debuger,* dan perangkat lain yang dipaket dalam Java Development Kit (JDK). Java2 adalah generasi yang sekarang sedang berkembang dari *platform*Java. Agar sebuah program Java dapat dijalankan, maka file dengan ekstensi ".java" harus dikompilasi menjadi file *bytecode*. Untuk menjalankan *bytecode* tersebut dibutuhkan JRE ( *Java Runtime*

*Environment*) yang memungkinkan pemakai untuk menjalankan program Java, hanya menjalankan, tidak untuk membuat kode baru lagi. JRE berisi JVM dan *library*Java yang digunakan.

*Platform*Java memiliki tiga buah edisi yang berbeda, yaitu J2EE ( *Java2 Enterprose Edition*), J2ME ( *Java2 Micro Edition*) dan J2SE ( *Java2 Second Edition*). J2EE adalah kelompok dari beberapa API ( *Application Programming Interface*) dari Java dan teknologi selain Java. J2EE sering dianggap sebagai *middleware* atau teknologi yang berjalan di *server*, namun sebenarnya J2EE tidak hanya terbatas untuk itu. Faktanya J2EE juga mencakup teknologi yang dapat digunakan di semua lapisan dari sebuah sistem informasi. Implementasi J2EE menyediakan kelas dasar dan API dari Java yang mendukung pengembangan dari rutin standar untuk aplikasi klien maupun *server*, termasuk aplikasi yang berjalan di *web browser*. J2SE adalah lingkungan dasar dari Java, sedangkan J2ME merupakan edisi *library* yang dirancang untuk digunakan pada *device* tertentu seperti *pagers* dan *mobile phone*.

Java merupakan bahasa pemrogrman yang bersifat *case sensitive* yang berarti penulisan menggunakan huruf besar ataupun huruf kecil pada kode program dapat berarti lain. Misalnya penulisan "System" akan diartikan berbeda dengan "system" oleh interpreter. Java tidak seperti C++, Java tidak mendukung pemrograman prosedural, tapi mendukung pemrograman berorientasi objek sehingga ada sintaks *class* pada kode programnya.

### 2.5.2. *Unified Modeling Language* (UML)

UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun system perangkat lunak . UML telah disahkan oleh OMG (*Object Management Group*) pada tahun 1997 sebagai standar *de-facto* dari pemodelan OOP dan diterima sebagai salah satu standar industri. Secara garis besar UML merupakan standard bahasa pemodelan untuk pembuatan *object-oriented* software dan merupakan kombinasi dari:

1. Konsep Pemodelan Data (*Entity Relationship Diagrams*)
2. Pemodelan Bisinis (*Work Flow*)
3. Pemodelan *Object*,
4. Pemodelan Komponen

Spesifikasi UML mendefinisikan sekumpulan diagram grafis sebagai tampilan dari beberapa level abstraksi dan UML dapat digunakan bersama oleh semua proses pada keseluruhan tahap siklus-hidup *(life-cycle)* pengembangan *software* serta pada implementasi ke beberapa teknologi yang berbeda.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram perangkat lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya, yaitu: Grady Booch OOD *(Object-Oriented Design)*, Jim Rumbaugh

OMT *(Object Modeling Technique)*, dan Ivar Jacobson OOSE *(Object-Oriented Software Engineering)*.

#### 2.5.2.1. Object Management Group (OMG)

OMG adalah konsorsium yang beranggotakan lebih dari 850 perusahaan untuk mendefinisikan standar-standar teknilogi objek termasuk CORBA (*Common Object Request Broker Architectur*) yang dibentuk pada tahun 1989 yang ditujukan untuk mempromosikan penggunaan teknologi berarah obyek pada aplikasi software.

*Unified Modeling Language* (UML) merupakan sistem arsitektur yang bekerja dalam OOAD (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software, dapat berupa model, deskripsi, atau software) yang terdapat dalam sistem software. UML merupakan bahasa pemodelan yang paling sukses dari tiga metode OO yang telah ada sebelumnya, yaitu Booch, OMT (*Object Modeling Technique*), dan OOSE (*Object-Oriented Software Engineering*).

UML merupakan kesatuan dari dari ketiga pemodelan tersebut dan ditambah kemampuan lebih karena mengandung metode tambahan untuk mengatasi masalah pemodelan yang tidak dapat ditangani ketiga metode tersebut. UML dikeluarkan oleh OMG (Object Management Group, Inc) yaitu organisasi internasional yang dibentuk pada 1989, terdiri dari perusahaan sistem informasi, *software developer*, dan para *user* sistem komputer.

### 2.5.2.2. Artifact UML

Diagram – diagram yang digunakan untuk mendefinisikan *UML* adalah sebagai berikut:

***a. Use Case Diagram***

Sebuah *use case* menggambarkan suatu urutan interaksi antara satu atau lebih aktor dan sistem. Dalam fase *requirements*, model *use case* mengambarkan sistem sebagai sebuah kotak hitam dan interaksi antara aktor dan sistem dalam suatu bentuk naratif, yang terdiri dari input user dan respon-respon sistem. Setiap use case menggambarkan perilaku sejumlah aspek sistem, tanpa mengurangi struktur internalnya. Selama pembuatan model use case secara pararel juga harus ditetapkan objek-objek yang terlibat dalam setiap use case.

*Use case* Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah deskripsi fungsi yang disediakan oleh sistem kedalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*.

*Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam sistem. Simbol yang digunakan yaitu:



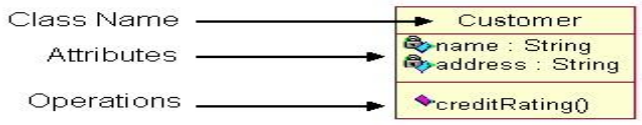

**Gambar 2.1** *Use Case* Model

Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

***b. Class Diagram***

Menggambarkan struktur statis *class* di dalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui berbagai cara: *associated* (terhubung satu sama lain), *dependent* (satu *class* tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya) atau *package* (grup bersama sebagai satu unit). Sebuah sistem biasanya mempunyai beberapa *class diagram.*

Suatu *class* biasanya terdiri dari 3 bagian, yaitu nama, atribut, dan operasi. Berikut adalah contoh dari suatu *class*:



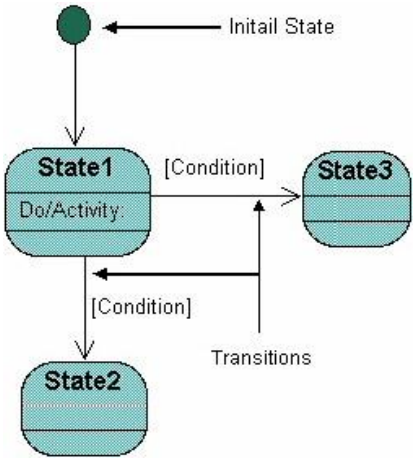

**Gambar 2.2** Bagian-bagian dari c*lass*

Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

### *c. State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan.

*State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.



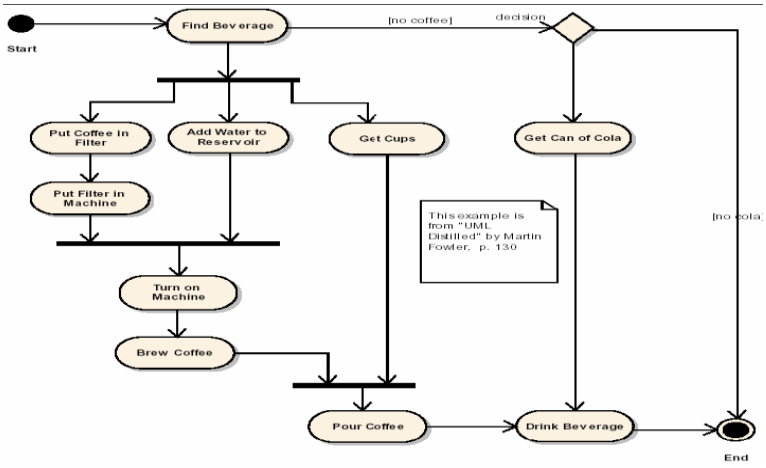

**Gambar 2.3** *State Diagram*

Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

### *d. Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama dengan *state*, standar *UML* menggunakan segi empat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses pararel (*fork on join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



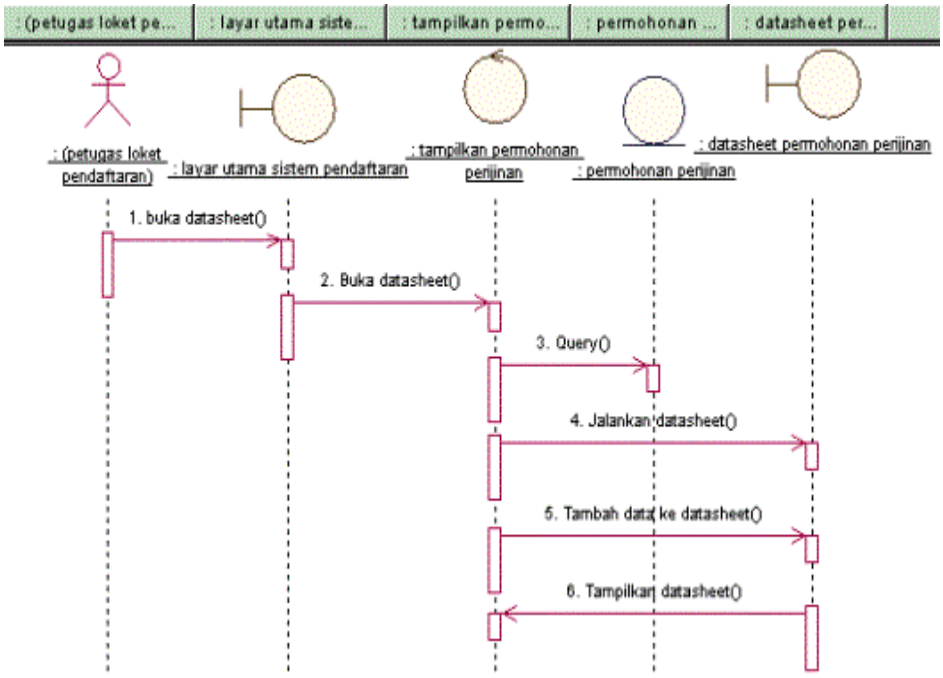

**Gambar 2.4** *Activity Diagram*

Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

### *e. Sequence Diagram*

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.

Komponen utama *sequence diagram* terdiri atas objek yang dituliskan dengan kotak segiempat bernama pesan diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan proses vertikal.




**Gambar 2.5** *Squence Diagram*

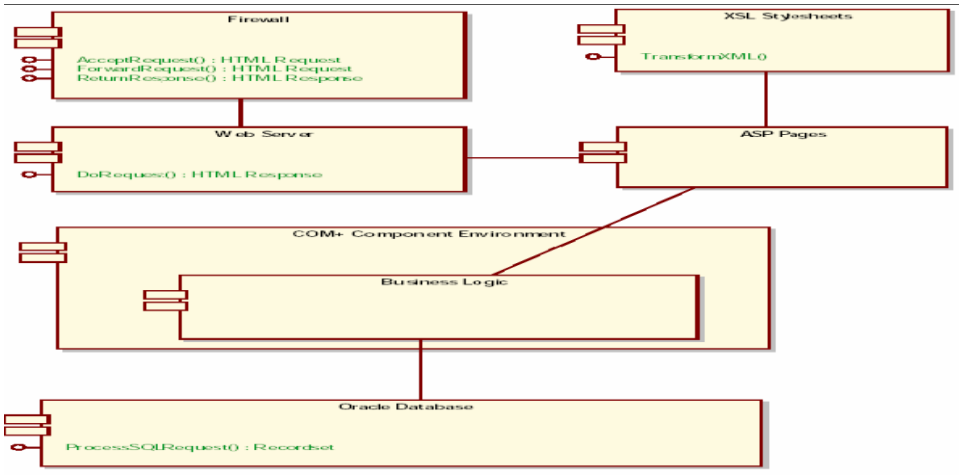
Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

### *f. Colaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration diagrams* menggambarkan object dan hubungannya (mengacu ke konteks). Jika penekanannya pada waktu atau urutan maka gunakan *sequence diagrams*, tapi jika penekanannya pada konteks gunakan *collaboration diagrams*.

### *g. Component Diagram*

Menggambarkan struktur fisik kode dari komponen. Komponen dapat berupa *source code*, komponen biner atau *executeable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view.*

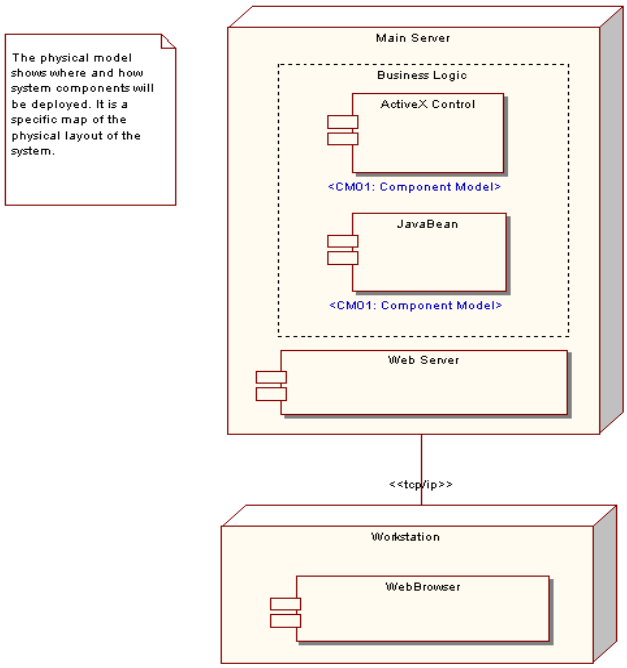



**Gambar 2.6** *Component Diagram*

Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

### *h. Deployment Diagram*

Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes, executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.




**Gambar 2.7** *Deployment Diagram*

Sumber [ http://www.scribd.com/doc/20728637/*Tutorial-UML-Dengan-Menggunakan-Rational-Rose/* 25 Juni 2011 ]

#### 2.5.2.3.Semantik Dalam UML

Seperti bahasa - bahasa lainnya, *UML* mendefinisikan notasi dan *syntax* / semantik. Notasi *UML* merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan *UML syntax* mendefinisikan bagaimana bentuk - bentuk tersebut dapat dikombinasikan. Notasi *UML* terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu *Grady Booch OOD (Object-Oriented Design)*, *Jim Rumbaugh OMT (Object Modeling Technique)*, dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering)* [http://www.IlmuKomputer.Com].

OMG telah menetapkan semantik (makna istilah) semua notasi *UML* dalam *Model Struktural* dan *Model Behavioral.*

1. *Model Struktural* atau model statis, menekankan pada struktur objek dalam sebuah sistem, menyangkut kelas - kelas, *interface*, *atribute* dan hubungan antar komponen.
2. *Model Behavioral* atau model dinamis, menekankan pada perilaku objek dalam sebuah sistem, menyangkut metode, interaksi, kolaborasi dan *state history* [http://unilanet.unila.ac.id].

#### 2.5.2.4.Tujuan UML

Adapun tujuan utama dari perancangan UML adalah [Har04] :

1. Menyediakan bahasa pemodelan visual yang ekspresif dan siap pakai untuk mengembangkan dan pertukaran model-model yang berarti.

2. Menyediakan mekanisme perluasan dan spesialisasi untuk memperluas konsep-konsep ini.
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untuk pemahaman bahasa pemodelan.
5. Mendorong pertumbuhan pasar kakas berorientasi objek.
6. Mendukung konsep-konsep pengembangan level lebih tinggi seperti komponen, kolaborasi, *framework* dan *pattern*.

**2.5.2.5.Cakupan UML**

Model *UML* dapat mencakup banyak perbendaharaan. Diantaranya adalah sebagai berikut :

1. *Things*
   a. *Structural*:
      1. *Use case*: deskripsi interaksi dengan *external actor*
      2. *Class*: deskripsi untuk objek - objek
      3. *Interface*: kumpulan operasi yang memberikan service tertentu untuk *class / component*
      4. *Component*: bagian sistem yang dapat diganti *(replaceable)* dan realisasinya sesuai dengan interface
   b. *Behaviour*
      1. Interaksi *(message sequence chart)*: pertukaran messages antar objek

2. *State machine*: urutan state dari objek dalam berinteraksi dengan objek lain

*c. Grouping*

*Package*: mekanisme untuk mengumpulkan elemen ke dalam satu set (group).

*d. Anotasi*

Catatan atau keterangan (teks) sebagai dokumentasi .

*2. Relationship*

*a. Dependency*

Hubungan antar element dimana perubahan pada elemen yang satu dapat mempengaruhi elemen yang lain *(dependent)*.

*b. Association*

Hubungan struktur antara elemen dan bertindak sebagai *link.*

*c. Generalization*

Hubungan dimana elemen yang special *(child)* mewarisi elemen yang umum *(parent).*

*d. Realization*

1. Hubungan (semantik) antara 2 elemen, dimana elemen yang satu memberikan kontrak dan elemen yang lain menjamin realisasi kontrak tersebut.
2. Dimana elemen yang special *(child)* mewarisi elemen yang umum *(parent)*

3. *Diagram*: *use case diagram*, *class diagram*, *msc diagram*, dan lain – lain

### 2.5.3. Microsoft Office Access 2007

Microsoft Access (atau Microsoft Office Access) adalah sebuah program aplikasi basis data komputer relasional yang ditujukan untuk kalangan rumahan dan perusahaan kecil hingga menengah. Aplikasi ini merupakan anggota dari beberapa aplikasi Microsoft Office, selain tentunya Microsoft Word, Microsoft Excel, dan Microsoft PowerPoint. Aplikasi ini menggunakan mesin basis data Microsoft Jet Database Engine, dan juga menggunakan tampilan grafis yang intuitif sehingga memudahkan pengguna.

Microsoft Access dapat menggunakan data yang disimpan di dalam format Microsoft Access, Microsoft Jet Database Engine, Microsoft SQL Server, Oracle Database, atau semua kontainer basis data yang mendukung standar ODBC. Para pengguna/*programmer* yang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang kompleks, sementara para programmer yang kurang mahir dapat menggunakannya untuk mengembangkan perangkat lunak aplikasi yang sederhana. Access juga mendukung teknik-teknik pemrograman berorientasi objek, tetapi tidak dapat digolongkan ke dalam perangkat bantu pemrograman berorientasi objek.

Access mengizinkan pengembangan yang relatif cepat karena semua tabel basis data, kueri, form, dan report disimpan di dalam berkas basis data miliknya

(*.MDB). Untuk membuat Query, Access menggunakan Query Design Grid, sebuah program berbasis grafis yang mengizinkan para penggunanya untuk membuat query tanpa harus mengetahui bahasa pemrograman SQL. DI dalam Query Design Grid, para pengguna dapat memperlihatkan tabel basis data sumber dari *query*, dan memilih *field-field* mana yang hendak dikembalikan oleh proses dengan mengklik dan menyeretnya ke dalam *grid*. *Join* juga dapat dibuat dengan cara mengklik dan menyeret field-field dalam tabel ke dalam *field* dalam tabel lainnya. Access juga mengizinkan pengguna untuk melihat dan memanipulasi kode SQL jika memang diperlukan.