BAB II

TINJAUAN PUSTAKA

II.1. Sistem

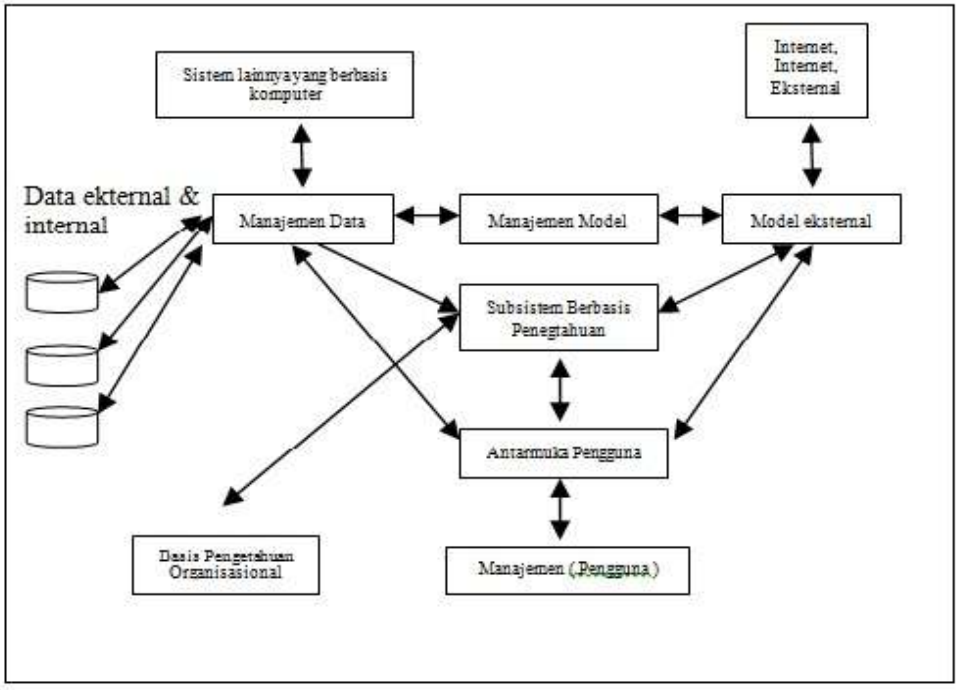
Sistem merupakan kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan dalam usaha mencapai suatu tujuan. Didalam perusahaan yang dimaksud elemen dari sistem adalah departemen-departemen internal, seperti persediaan barang mentah, produksi, persedian barang jadi, promosi, penjualan, keuangan, personalia serta pihak eksternal seperti *supplier* dan konsumennya yang saling terkait satu sama lain dan membentuk satu kesatuan usaha (Budi Sutejo Dharma Oetomo; 2006: 168).

II.1.1. Sistem Pendukung Keputusan (*Decision Support System*)

Sistem Pendukung Keputusan merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semi terstruktur dan situasi yang tidak terstruktur, dimana tidak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat (Alter; 2002). DSS biasanya dibangun untuk mendukung solusi atau suatu masalah atau untuk mengevaluasi suatu peluang. DSS yang seperti itu disebut aplikasi DSS. Aplikasi DSS digunakan dalam mengambil keputusan. Aplikasi DSS menggunakan CBIS (*Computer Based Infomation System*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktur.

Aplikasi DSS menggunakan data, memberikan antarmuka pengguna yang mudah, dan dapat menggabungkan pemikiran pengambil keputusan. DSS lebih ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analisis dalam situasi yang kurang terstruktur dan dengan kriteria yang kurang jelas. DSS dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambilan keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia (Kusrini; 2007: 15).

Tabel II.1 Arsitektur DSS



Sumber : Kusrini (2007 : 26)

II.1.2. Metode *Decision Tree*

Decision Tree adalah sebuah struktur pohon, dimana setiap *node* pohon memepresentasikan atribut yang telah diuji, setiap cabang merupakan suatu pembagian hasil uji, dan *node* daun (*leaf*) merepresentasikan kelompok kelas tertentu. Level *node* teratas dari sebuah *decision tree* adalah *node* akar (*root*) yang biasanya berupa atribut yang paling memiliki pengaruh terbesar pada suatu kelas tertentu.

Iterative Dichotomicer 3 (ID3) adalah algoritma *decision tree learning* (algoritma pembelajaran pohon keputusan) yang paling dasar. Algoritma ini melakukan pencarian secara rakus /menyeluruh (*greedy*) pada semua kemungkinan pohon keputusan (Wahyudin, 2009:6). Salah satu algoritma induksi pohon keputusan yaitu ID3 (*Iterative Dichotomiser 3).* ID3 dikembangkan oleh J. Ross Quinlan.

Algoritma ID3 dapat diimplementasikan menggunakan fungsi *rekursif* (fungsi yang memanggil dirinya sendiri). Algoritma ID3 berusaha membangun *decision tree* (pohon keputusan) secara *top-down* (dari atas ke bawah), mulai dengan pertanyaan : “atribut mana yang pertama kali harus dicek dan diletakkan pada *root*?” pertanyaan ini dijawab dengan mengevaluasi semua atribut yang ada dengan menggunakan suatu ukuran statistic (yang banyak digunakan adalah *information gain*) untuk mengukur efektivitas suatu atribut dalam mengklasifikasikan kumpulan sampel data.

Berikut rumus dari *decision tree :*

$$Entropy S) = -P_{Yes} \log_2 P_{Yes} - P_{No} \log_2 P_{No}$$

Rumus mencari *information gain* dari *decision tree :*

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

$$E(A) = \sum_{i=1}^{v}\frac{p_i + n_i}{p+n} I(p_i, n_i)$$

$$Gain(A) = I(p,n) - E(A)$$

II.2. UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah suatu alat Bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek (Munawar ; 2005 : 17). Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain. Meskipun UML sudah banyak menyediakan diagram yang bisa membantu mendefenisikan suatu aplikasi, tidak berarti bahwa semua diagram tersebut akan bisa menjawab persoalan yang ada. Adapun tipe diagram UML yang ada seperti pada Tabel II.1.

Tabel II.2 Tipe Diagram UML

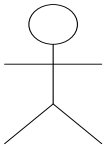
Diagram	Tujuan	Keterangan
Activity	Prilaku prosedural dan parallel	Sudah ada di UML 1
Class	Class, fitur dan relasinya	Sudah ada di UML 1
Communication	Interaksi diantara objek. Lebih menekankan kepada link	Di UML 1 disebut collaboration
Component	Struktur dan koneksi dari komponen	Sudah ada di UML 1
Composite Structure	Dekomposisi sebuah class saat runtime	Baru untuk UML 2
Deployment	Penyebaran/instalasi ke klien	Sudah ada di UML 1
Interaction Overview	Gabungan dari activity dan sequence diagram	Baru untuk UML 1
Object	Contoh konfigurasi instance	Tidak resmi ada di UML 1
Package	Struktur hierarki saat kompilasi	Tidak resmi ada di UML 1
Sequence	Interaksi antara objek. Lebih menekankan pada urutan.	Sudah ada di UML 1
State Machine	Bagaimana event mengubah sebuah objek	Sudah ada di UML 1
Timing	Interaksi antar objek. Lebih menekankan pada waktu	Sudah ada di UML 1
Use Case	Bagaimana user berinteraksi dengan sebuah sistem	Sudah ada di UML 1

Sumber : "(Munawar ; 2005 : 23)"

II.2.1. Notasi Dasar UML

1. Actor

Actor adalah *abstraction* dari orang dan *system* yang lain yang mengaktifkan fungsi dari target *system*. Orang atau *system* bisa muncul dalam bebrapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. Berikut notasi *actor* dalam UML:



Gambar II.1 : Notasi *Actor* pada UML

Sumber : " (Munawar ; 2005 : 64)"

2. Class

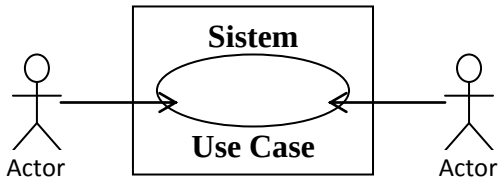
Class, dalam notasi UML digambarkan dengan kotak. Nama class menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak. Bila *class* mempunyai nama yang terdiri dari 2 suku kata atau lebih, maka semua suku kata digabungkan tanpa spasi dengan huruf awal tiap suku kata menggunakan huruf besar. Berikut notasi *class* dalam UML:

Gambar II.2 : Notasi *Class* di UML

Sumber : " (Munawar ; 2005 : 35)"

3. Use Case

Use Case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu *system* dari sudut pndangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah *system*. Karena *system* pengembangan tradisional sering ceroboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu. Notasi *use cas*e dapat dilihat pada gambar II.5 :



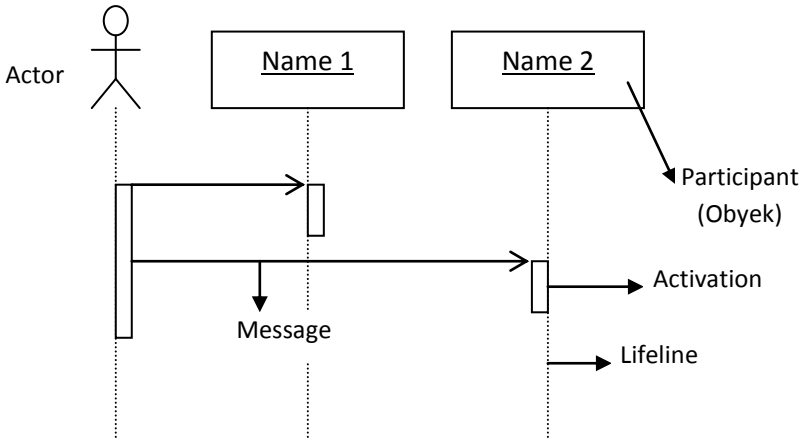
Gambar II.3 : Notasi *Use Case* pada UML

Sumber : " (Munawar ; 2005 : 64)"

4. Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini dalam *use case*.

Komponen utama squence diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. *Message* dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical. Berikut Contoh *sequence diagram* :



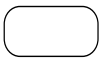
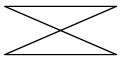
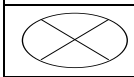
Gambar II.4 : Simbol-simbol *Sequence Diagram*

Sumber : " (Munawar ; 2005 : 89)"

5. Activity Diagram

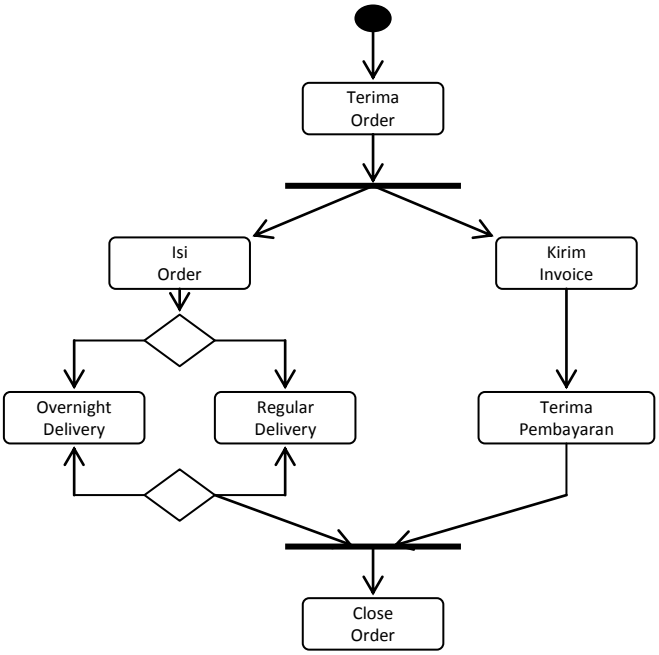
Activity diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung prilaku paralel sedangkan *flowchart* tidak bisa. Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram*.

Tabel II.3 Simbol-simbol *Activity Diagram*

Simbol	Keterangan
	Titik awal
	Titik akhir
	Activity
	Pilihan untuk pengambilan keputusan
	Fork; digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	Rake; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (Flow Final)

Sumber : "(Munawar ; 2005 : 109)"

Adapun contoh dari *Activity Diagram* dapat di lihat pada Gambar II.8.



Gambar II.5 : Contoh *Activity Diagram* Sederhana

Sumber : " (Munawar ; 2005 : 111)"

II.3. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan SI, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun-menyusun informasi-informasi dalam berbagai bentuk. *Database* merupakan himpunan kelompok data yang saling berkaitan. Data tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi yang tidak perlu, sehingga dapat diolah atau di eksplorasi secara cepat dan mudah untuk menghasilkan informasi (Budi Sutedjo Dharma Oetomo ; 2006: 99).

II.3.1. Hierarki Data Dalam Database

Data dalam sebuah *database* disusun berdasarkan sistem hierarki yang unik, yaitu:

1. *Database*, merupakan kumpulan file yang saling terkait satu sama lain, misalnya file data induk karyawan, file jabatan file penggajian dan lain sebagainya. Kumpulan file yang tidak saling terkait satu sama lain tidak dapat disebut *database*, misalnya file data induk karyawan, file tamu undangan perkawinan, file barang retail pasar swalayan.
2. *File*, yaitu kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. *Record*, yaitu kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. *Field*, yaitu atribut dari *record* yang menunjukkan suatu item dari data, seperti nama, alamat, dan lain sebagainya.
5. *Byte*, yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus
6. *Bit*, yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte* (Budi Sutedjo Dharma Oetomo; 2006: 102).

II.3.2. MySQL

Database MySQL merupakan system manajemen basis data SQl yang sangat terkenal dan bersifat open source, meski dirilis secara open source namun

kehandalannya dapat dibuktikan. MySQL mempunyai stabilitas dan kecepatan akses yang tinggi, dapat berjalan pada berbagai system operasi, mudah digunakan, dan tersedia dalam berbagai macam bahasa. MySQL dikembangkan untuk menangani database yang besar secara cepat dan telah sukses digunakan selama bertahun-tahun. Konektivitas, kecepatan, dan keamananya telah menjadikan MySQL sebagai server yang cocok untuk mengakses database di internet. (Budi Raharjo: 2009: 21) .

II.4. Kamus Data

Kamus data ikut berperan dalam perancangan dan pembangunan SI karena peralatan ini berfungsi untuk :

1. Menjelaskan arti aliran data dan penyimpanan dalam penggambaran dalam data flow diagram.
2. Mendeskripsikan komposisi paket data yang bergerak melalui aliran, misalnya data alamat diurai menjadi nama jalan, nomor, kota, Negara dan kode pos.
3. Menjelaskan spesifikasi nilai dan satuan yang relevan terhadap data yang mengalir dalam sistem tersebut (Budi Sutejo Dharma Oetomo; 2006: 118).

Tabel II.4 Notasi Kamus Data

Simbol	Uraian
=	Terdiri atas, mendefinisikan, diuraikan menjadi, artinya Contoh: nama=sebutan+nama1+nama2+gelar1+gelar2
+	Dan
( )	Optional (pilihan boleh ada atau boleh tidak) Contoh: alamat=alamat rumah+(alamat surat)
|	Sama dengan simbol []
{ }	Pengulangan Contoh: nama1={karakter_valid}

[ ]	Memilih salah satu dari sejumlah alternatif, seleksi Contoh: sebutan = [Bapak| Ibu| Yang mulia]
**	Komentar Contoh: *seminar yang akan diikuti*
|	Pemisah sejumlah alternatif pilihan antara simbol [ ]

Sumber : ”(Budi Sutejo Dharma Oetomo; 2006: 118).”

II.5. *Entity Relationship Diagram* – ERD

II.5.1. Model-model Data

Struktur yang mendasari suatu basisdata adalah model data yang merupakan kumpulan alat-alat konseptual untuk mendeskripsikan data, relasi data, data semantik, dan batasan konsistensi. Untuk mengilustrasikan konsep model data, berikut disajikan dua model data, yaitu *entity relationship model* dan *relational model.* Kedua model menyediakan cara mendeskripsikan rancangan basisdata pada tingkatan logis.

II.5.2. *Entity Relationship Model*

Entity Relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat pula dianggap sebagai entitas.

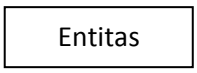
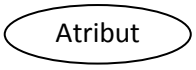
Entitas digambarkan dalam basisdata dengan kumpula atribut. Misalnya atribut nim, nama, alamat dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan entitas mata kuliah.

Atribut NIM digunakan untuk mengidentifikasi mahasiswa secara unik karena dimungkinkan terhadap dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh, relasi menghubungkan mahasiswa dengan mata kuliah yang di ambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas (*entity set)*, sedangkan kumpulan semua relasi bertipe sama disebut kumpulan relasi (*relationship set)*.

Struktur logis (skema *database*) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut :

Tabel II.5 Notasi ERD *(Entity Relationship Diagram)*

	Persegi panjang mewakili kumpulan entitas
	Elips mewakili atribut
	Belah ketupat mewakili relasi
	Garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

Sumber : "(Janner Simarmata & Imam Prayudi: 2006: 59)"

II.6. 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 relational.

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 (Janner Simarmata & Imam Prayudi; 2006: 77).

1. Bentuk Normal Pertama (1NF/*First Normal Form*), bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak (*multivalues atribute*) telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom satu nilai untuk irisan baris dan kolom pada tabel.
2. Bentuk Normal Kedua (2NF/*Second Normal Form*), semua kebergantungan fungsional (*functional dependeny*) yang bersifat sebagian (*partial functional dependeny*) telah dihilangkan.
3. Bentuk Normal Ketiga (3NF/*Third Normal Form*), semua kebergantungan transitif (*transitive dependeny*) telah dihilangkan.
4. Boyce-Codd Normal Form (BCNF/*Boyce-Codd Normal Form*), semua anomali yang tersisa dari hasil penyempurnaan kebergantungan fungsional (*functional dependeny*) diatas telah dihilangkan.
5. Bentuk Normal Keempat (4NF/*Fifth Normal Form*), semua anomali yang berasal dari kebergantungan banyak-nilai (*multivalues dependeny*) telah dihilangkan (Adi Nugroho; 2010: 34).

Tujuan normalisasi adalah membuat kumpulan tabel relasional yang bebas dari data berulang yang dapat dimodifikasi secara benar dan konsisten. Ini berarti bahwa semua tabel pada basisdata relasional harus berada pada bentuk normal ketiga (3NF). Sebuah tabel relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah (a) saling independen dan (b) sepenuhnya tergantung pada kunci utama. Saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada senbarang kombinasi kolom lainnya. Dua bentuk normal pertama adalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua tabel dalam 3NF (Stephens and Plew, 2000) (Janner Simarmata & Imam Prayudi; 2006: 77).

II.7. Bahasa Pemrograman Java

Java merupakan bahasa pemrograman berorientasi objek dan bebas *platform*, dikembangkan oleh SUN *Micro System* dengan sejumlah keunggulan yang memungkinkan Java dijadikan sebagai bahasa pengembangan *enterprise* (Rijalul Fikri, dkk; 2005: 15).

Java lahir karena ketidakpuasan seorang insinyur di SUN *Micro System* bernama James Gosling. Ia tidak puas dengan kompiler C++ (yang ia gunakan untuk membuat *software* yang di-*embed* pada peralatan elektronik) karena dinilai terlalu banyak menghasilkan *bug*, berbiaya besar, sangat bergantung terhadap *platform.* Gosling merasa perlu membuat kompiler baru sebagai solusi terhadap sejumlah kelemahan pada C++ tersebut.

Kompiler baru tersebut diberi nama dengan Oak. Kompiler ini mirip dengan C++ tetapi dengan sejumlah pengurangan fitur yang dianggap kurang

menguntungkan dalam pengembangan, seperti *multipel inheritance*, konversi tipe secara otomatis, penggunaan pointer dan manajemen memori (Rijalul Fikri, dkk; 2005: 19).

Beberapa istilah penting Java yang perlu diketahui:

1. *Java Developers Kit* (JDK), merupakan kumpulan utilitas yang diperlukan untuk menggkompilasi program Java.
2. *Java Software Development Kit* (J2SDK), merupakan gabungan dari JDK dan utilitas untuk menjalankan program Java pada PC.
3. *The Java API*, merupakan kumpulan *library* pada java yang menyediakan modul-modul generik yang diperlukan *programmer* selama pengembangan.

Salah satu alasan mengapa Java tepat digunakan untuk menangani kebutuhan aplikasi *enterprise* adalah keampuhannya dalam soal keamanan. Fitur keamanan Java ada 2 paket, yaitu pada JDK dan pada Java *Cryptography Extension* (JCE). Fitur keamanan yang tersedia pada kedua paket meliputi *signature* (untuk mendatangi dokumen), *message digest*, pembangkitan kunci, *autentikasi*, *enskripsi*, dan bilangan besar (Rijalul Fikri, dkk; 2005: 18).

Berikut ini adalah gambar pemrograman Java :