BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Model perancangan sesungguhnya adalah modal objek yang mendeskripsikan realisasi fisik use case dengan cara berfokus pada bagaimana spesifikasi-spesifikasi kebutuhan fungsional dan non-fungsional, bersama dengan batasan-batasan lain yang berhubungan dengan lingkungan implemenatasi, memiliki imbas langsung pada pertimbangan-pertimbangan pada aktivitas-aktivitas yang dilakukan pada tahap implementasi. Tambahannya, model perancangan sesungguhnya secara langsung bertindak sebagai abstraksi implementasi sistem/perangkat lunak dan dengan sendirinya model perancangan suatu saat nanti akan menjadi asupan bagi aktivitas-aktivitas selanjutnya yang kelak akan terdefinisi pada tahap implementasi (Adi Nugroho ; 2010 : 212).

II.2. Pengertian Aplikasi

Program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi *software* yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

a. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu (Rahmatillah ; 2011: 3).

II.3. Keamanan Data

Keamanan sebuah informasi merupakan suatu hal yang harus diperhatikan. Masalah tersebut penting karena jika sebuah informasi dapat di akses oleh orang yang tidak berhak atau tidak bertanggung jawab, maka keakuratan informasi tersebut akan diragukan, bahkan akan menjadi sebuah informasi yang menyesatkan.

Ada banyak cara mengamankan data atau informasi pada sebuah sistem. Pada umumnya pengamanan data dapat dikategorikan menjadi dua jenis, yaitu : penecegahan (*presentif*) dan pengobatan (*recovery*). Pencegahan dilakukan supaya data tidak rusak, hilang dan dicuri, sementara pengobatan dilakukan apabila data sudah terkena virus, sistem terkena worm, dan lubang keamanan sudah *diexploitasi*. Sistem keamanan informasi (*information security*) memiliki empat tujuan yang sangat mendasar adalah :

1. Kerahasiaan (*Confidentiality*). Informasi pada sistem komputer terjamin kerahasiaannya, hanya dapat diakses oleh pihak-pihak yang diotorisasi, keutuhan serta konsistensi data pada sistem tersebut tetap terjaga. Sehingga upaya orang-orang yang ingin mencuri informasi tersebut akan sia-sia.
2. Ketersediaan (*Availability*). Menjamin pengguna yang sah untuk selalu dapat mengakses informasi dan sumberdaya yang diotorisasi. Untuk memastikan bahwa orang-orang yang memang berhak untuk mengakses informasi yang memang menjadi haknya.

3. Integritas (*Integrity*) Menjamin konsistensi dan menjamin data tersebut sesuai dengan aslinya, sehingga upaya orang lain yang berusaha merubah data akan segera dapat diketahui.
4. Penggunaan yang sah (*Legitimate Use*). Menjamin kepastian bahwa sumberdaya tidak dapat digunakan oleh orang yang tidak berhak (Paryati ; 2012 : 379).

II.3.1. Masalah dalam Keamanan Data

Ancaman terhadap sistem informasi dibagi menjadi 2 macam, yaitu ancaman aktif dan ancaman pasif.

a. Ancaman aktif mencakup :
    1. Pencurian data Jika informasi penting yang terdapat dalam database dapat diakses oleh orang yang tidak berwenang maka hasilnya dapat kehilangan informasi atau uang. Misalnya, mata-mata industri dapat memperoleh informasi persaingan yang berharga, penjahat komputer dapat mencuri uang bank.
    2. Penggunaan sistem secara ilegal Orang yang tidak berhak mengakses informasi pada suatu sistem yang bukan menjadi hak-nya, dapat mengakses sistem tersebut. Penjahat komputer jenis ini umumnya adalah hacker yaitu orang yang suka menembus sistem keamanan dengan tujuan mendapatkan data atau informasi penting yang diperlukan, memperoleh akses ke sistem telepon, dan membuat sambungan telepon jarak jauh secara tidak sah.
    3. Penghancuran data secara ilegal Orang yang dapat merusak atau menghancurkan data atau informasi dan membuat berhentinya suatu sistem operasi komputer. Penjahat komputer ini tidak perlu berada ditempat kejadian. Ia dapat masuk melalui jaringan komputer dari suatu terminal dan menyebabkan kerusakan pada semua sistem dan

hilangnya data atau informasi penting. Penjahat komputer jenis ini umumnya disebut sebagai cracker yaitu penjebol sistem komputer yang bertujuan melakukan pencurian data atau merusak sistem.

4. Modifikasi secara ilegal Perubahan-perubahan pada data atau informasi dan perangkat lunak secara tidak disadari. Jenis modifikasi yang membuat pemilik sistem menjadi bingung karena adanya perubahan pada data dan perangkat lunak disebabkan oleh progam aplikasi yang merusak (malicious *software*). Program aplikasi yang dapat merusak tersebut terdiri dari program lengkap atau segemen kode yang melaksanakan fungsi yang tidak dikehendaki oleh pemilik sistem. Fungsi ini dapat menghapus file atau menyebabkan sistem terhenti. Jenis aplikasi yang dapat merusak data atau perangkat lunak yang paling populer adalah virus.

b. Ancaman pasif mencakup :

1. Kegagalan sistem Kegagalan sistem atau kegagalan *software* dan hardware dapat menyebabkan data tidak konsisten, transaksi tidak berjalan dengan lancar sehingga data menjadi tidak lengkap atau bahkan data menjadi rusak. Selain itu, tegangan listrik yang tidak stabil dapat membuat peralatan-peralatan menjadi rusak dan terbakar.
2. Kesalahan manusia Kesalahan pengoperasian sistem yang dilakukan oleh manusia dapat mengancam integritas sistem dan data.
3. Bencana alam Bencana alam seperti gempa bumi, banjir, kebakaran, hujan badai merupakan faktor yang tidak terduga yang dapat mengancam sistem informasi sehingga mengakibatkan sumber daya pendukung sistem informasi menjadi luluhlantah dalam waktu yang singkat (Paryati ; 2012 : 380)

II.3.2. Mengamankan Sistem

Ada banyak cara mengamankan data atau informasi pada sebauh sistem. Pada umumnya pengamanan data dapat dikategorikan menjadi dua jenis, yaitu : penecegahan (*presentif*) dan pengobatan (*recovery*).

1. Pengendalian akses. Pengendalian akses dapat dicapai dengan tiga langkah, yaitu :
    a. Identifikasi pemakai (*user identification*). Mula-mula pemakai mengidentifikasikan dirinya sendiri dengan menyediakan sesuatu yang diketahuinya, seperti kata sandi atau password. Identifikasi tersebut dapat mencakup lokasi pemakai, seperti titik masuk jaringan dan hak akses telepon.
    b. Pembuktian keaslian pemakai (*user authentication*). Setelah melewati identifikasi pertama, pemakai dapat membuktikan hak akses dengan menyediakan sesuatu yang ia punya, seperti kartu id (*smart card, token* dan *identification chip*), tanda tangan, suara atau pola ucapan.
    c. Otorisasi pemakai (*user authorization*). Setelah melewati pemeriksaan identifikasi dan pembuktian keaslian, maka orang tersebut dapat diberi hak wewenang untuk mengakses dan melakukan perubahan dari suatu file atau data.
2. Memantau adanya serangan pada system

    Sistem pemantau (*monitoring system*) digunakan untuk mengetahui adanya penyusup yang masuk kedalam sistem (*intruder*) atau adanya serangan (*attack*) dari *hacker*. sistem ini biasa disebut "*intruder detection system*" (IDS). Sistem ini dapat memberitahu admin melalui e-mail atau melalui mekanisme lain. Terdapat berbagai cara untuk memantau adanya

penyusup. Ada yang bersifat aktif dan pasif. IDS cara yang pasif misalnya dengan melakukan pemantauan pada logfile. Berbagai macam *software* IDS antara lain, yaitu :

a. *Autobuse* yaitu mendeteksi *port scanning* dengan melakukan pemantauan pada *logfile*.
b. *Port blocker* yaitu memblok port tertentu terhadap serangan. Biasanya untuk melakukan port blok memerlukan *software* tertentu, seperti NinX atau sejenisnya.
c. *Courtney* dan *portsentry* yaitu mendeteksi *port scanning* dengan melakukan pemantauan paket data yang sedang lewat.
d. *Snort* yaitu mendeteksi pola pada paket data yang lewat dan mengirimkan instruksi siaga jika pola tersebut terdeteksi. Pola disimpan dalam berkas yang disebut library yang dapat dikonfigurasi sesuai dengan kebutuhan.

3. Penggunaan enkripsi

Salah satau mekanisme untuk meningkatkan keamanan sistem yaitu dengan menggunakan teknologi enkripsi data. Data-data yang dikirimkan diubah sedemikian rupa sehingga tidak mudah diketahui oleh orang lain yang tidak berhak. Ada tiga kategori enkripsi yaitu :

a. Enkripsi rahasia. Terdapat sebuah kunci yang dapat digunakan untuk meng-enkripsi dan men-dekripsi data data.
b. Enkripsi publik. Terdapat dua kunci yang digunakan, satu kunci digunakan untuk melakukan enkripsi dan kunci yang lain digunakan untuk melakukan proses dekripsi.
c. Fungsi one-way. Suatu fungsi dimana informasi di enkripsi untuk menciptakan “signature” dari data asli yang dapat digunakan untuk keperluan autentifikasi. Enkripsi dibentuk berdasarkan algoritma yang dapat mengacak data kedalam bentuk yang tidak bisa dibaca atau rahasia, sedangkan dekripsi dibentuk berdasarkan algoritma yang sama untuk mengembalikan data yang teracak menjadi bentuk asli atau dapat dibaca.

Metode enkripsi Ada beberapa metode enkripsi yaitu :

a) DES (*Data Encryption Standard*) DES merupakan nama dari sebuah algoritma untuk mengenkripsi data yang dikeluarkan oleh *Federal Information Processing Stadard* (FIPS) Amerika Serikat. DES memiliki blok kunci 64-bit, tetapi yang digunakan dalam proses eksekusi adalah 54 bit. Algoritma enkripsi ini termasuk algoritma yang tidak mudah untuk diterobos.

b) 3DES (*Triple* DES) Triple DES dikembangkan untuk mengatasi kelemahan ukuran kunci yang digunakan pada proses enkripsi-deskripsi DES sehingga teknik kriptografi ini lebih tahan terhadap *exhaustive key* search yang dilakukan oleh kriptoanalis. Penggunaan *triple* DES dengan suatu kunci tidak akan menghasilkan pemetaan yang sama seperti yang dihasilkan oleh DES dengan kunci tertentu. Hal itu disebabkan oleh sifat DES yang tidak tertutup (*not closed*). Sedangkan dari hasil implementasi dengan menggunakan modus Electronic *Code Book* (ECB) menunjukkan bahwa walaupun memiliki kompleksitas atau notasi O yang sama (O(n)), proses enkripsi-deskripsi pada DES lebih cepat dibandingkan dengan triple DES.

c) *Kerberos*. Kerberos adalah suatu sistem keamanan berdasarkan enkripsi yang menyediakan pembuktuan keaslian (mutual authentication) bersama-sama antara komponen client dan komponen *server* dalam lingkungan computing terdistribusi. Kerberos juga menyediakan hak-hak layanan yang dapat digunakan untuk mengontrol client mana yang berwenang mengakses suatu *server*.

4. Melakukan backup secara rutin.

Dengan adanya backup data yang dilakukan secara rutin merupakan sebuah hal yang esensial, sehingga apabila ada penyusup yang mencuri, menghapus, bahkan melakukan modifikasi seluruh isi berkas penting dapat diatasi dengan cepat (Paryati ; 2012 : 382).

II.4. Pengertian Google Drive

Google Drive adalah layanan *cloud storage* dari *Google* yang diluncurkan pada akhir April 2012, yaitu layanan untuk menyimpan file di internet pada storage yang disediakan oleh *Google*. Dengan menyimpan file di *Google Drive* maka pemilik file dapat mengakses *file* tersebut kapanpun dimanapun dengan menggunakan komputer desktop, laptop, komputer tablet ataupun smartphone. Dan file tersebut dapat di share dengan orang lain untuk berbagi pakai dan juga kolaborasi peng-edit-annya. Kapasitas yang disediakan oleh *google* drive untuk layanan gratis adalah 5GB, untuk menggunakan kapasitas lebih dari itu maka akan dikenakan biaya tambahan.

Dengan menggunakan *Google Drive*, berarti pemilik file telah memiliki back-up file nya di internet sehingga jika terjadi sesuatu pada file yang disimpan di komputer atau laptop, misalnya file tersebut rusak atau hilang atau terkena virus, atau komputer/laptopnya rusak yang menyebabkan tidak dapat digunakan, maka file yang berada di *Google Drive* tetap aman dan tetap dapat diakses menggunakan komputer lain yang terhubung ke internet (http://www.cloudindonesia.or.id/mengenal-menggunakan-*google*-drive.html)

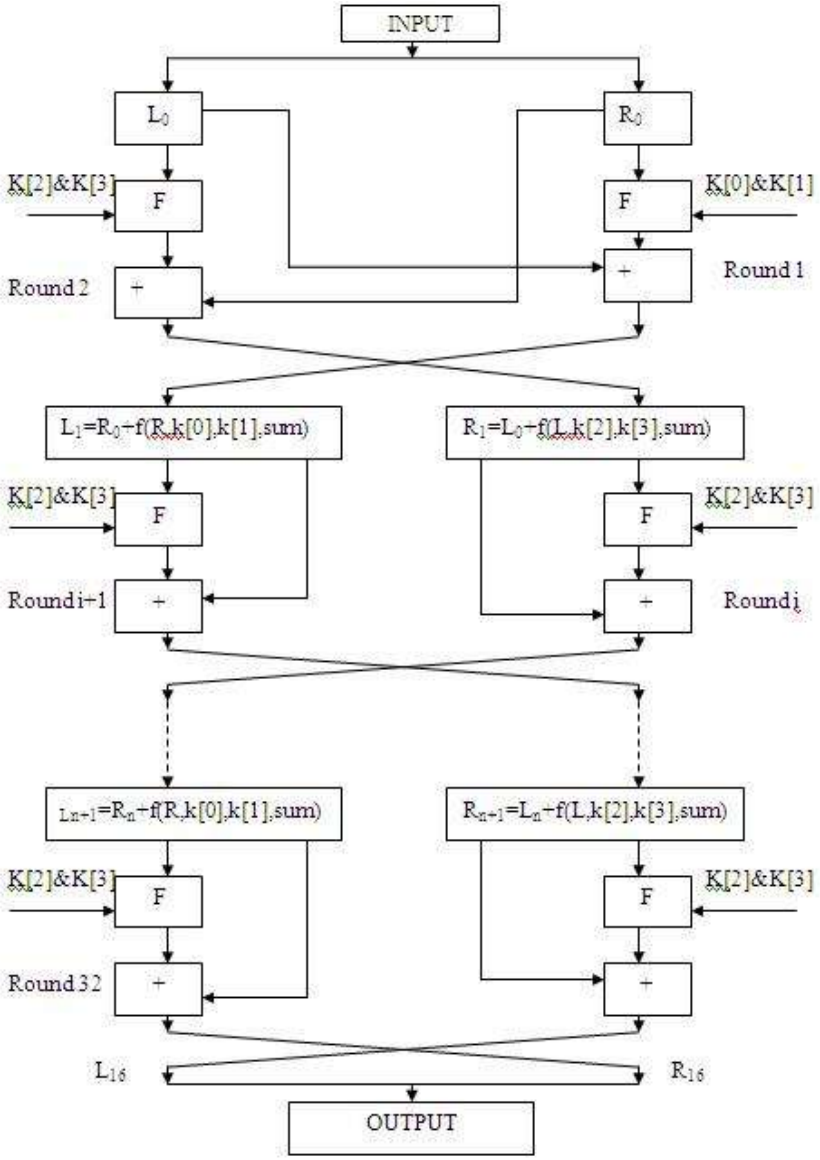
II.5. Kriptografi TEA (*Tiny Encryption Algorithm*)

Tiny Encryption Algorithm (TEA) merupakan suatu algoritma sandi yang diciptakan oleh David J. Wheeler dan Roger M. Needham dari Cambridge University tahun 1994. Algoritma ini

merupakan algoritma penyandian block cipher yang dirancang untuk penggunaan memori yang seminimal mungkin dengan kecepatan proses yang maksimal.

Sistem penyandian TEA menggunakan proses feistel network dengan menambahkan fungsi matematik berupa penambahan dan pengurangan sebagai operator pembalik selain XOR. Hal ini dimaksudkan untuk menciptakan sifat non-linearitas. Pergeseran dua arah (ke kiri dan ke kanan) menyebabkan semua bit kunci dan data bercampur secara berulang ulang.

TEA memproses 64-bit input sekali waktu dan menghasilkan 64- bit output. TEA menyimpan 64-bit input kedalam L0 dan R0 masing masing 32-bit, sedangkan 128-bit kunci disimpan kedalam k[0], k[1], k[2], dan k[3] yang masing masing berisi 32-bit. Diharapkan teknik ini cukup dapat mencegah penggunaan teknik exshautive search secara efektif. Hasil outputnya akan disimpan dalam L16dan R16. Bilangan delta konstan yang digunakan adalah 9E3779B9, dimana bilangan delta berasal dari golden number, digunakan delta = (√5 -1)231 . Suatu bilangan delta ganda yang berbeda digunakan dalam setiap roundnya sehingga tidak ada bit dari perkalian yang tidak berubah secara teratur. Berbeda dengan struktur feistel yang semula hanya mengoperasikan satu sisi yaitu sisi sebelah kanan dengan sebuah fungsi F, pada algoritma TEA kedua sisi dioperasikan dengan sebuah fungsi yang sama.



Gambar II.1. Enkripsi Data

Untuk melakukan enkripsi, Proses diawali dengan *input-bit* teks sebanyak 64-bit, kemudian 64-bit teks tersebut dibagi menjadi dua bagian, yaitu sisi kiri (L0) sebanyak 32-bit dan sisi kanan (R0) sebanyak 32-bit. Setiap bagian teks akan dioperasikan sendiri-sendiri. R0 (Z) akan digeser kekiri sebanyak empat (4) kali dan ditambahkan dengan kunci k[0], sementara itu Z ditambah dengan sum (delta) yang merupakan konstanta. Hasil penambahan ini di-XOR-kan dengan penambahan sebelumnya. Langkah selanjutnya di-XOR-kan dengan hasil penambahan antara Z

yang digeser kekanan sebanyak lima (5) kali dengan kunci k[1]. Hasil tersebut kemudian ditambahkan dengan L0 (Y) yang akanmenjadi R1 (Mukti Qamal ; 2014 : 22).

II.5.1. Jenis – Jenis Kriptografi TEA

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya :

1. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya)

   Algoritma ini memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Bila mengirim pesan dengan menggunakan algoritma ini, si penerima pesan harus diberitahukan kunci dari pesan tersebut agar bisa mendekripsikan pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan.

2. Algoritma Asimetri (menggunakan kunci yang berbeda untuk enkripsi dan dekripsi)

   Algoritma kunci-simetri mengacu pada metode enkripsi yang dalam hal ini baik pengirim maupun penerima memiliki kunci yang sama. Algoritma kunci-simetri modern beroperasi dalam mode bit dan dapat dikelompokkan menjadi dua kategori, yaitu :

   a. Cipher Aliran (*Stream Cipher*) Algoritma kriptografi beroperasi dalam plaintexs/ciphertext dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsi/didekripsikan bit per bit. Chiper aliran mengenkripsi satu bit pada setiap kali proses enkripsi. 2.
   b. Cipher Blok (*Block Cipher*) Algoritma kriptografi beroperasi pada plaintexs/ciphertext dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi dalam blokblok bit yang panjangnya sudah ditentukan sebelumnya. Misalnya panjang blok adalah 64 bit, maka

itu berarti algoritma enkripsi memperlakukan 8 karakter setiap kali enkripsi (1 karakter = 8 bit dalam pengkodean ASCII). Cipher blok mengenkripsi satu blok bit setiap kali.

3. *Hash Function*

Plaintext dibagi menjadi beberapa blok dengan panjang tetap. Beberapa mode operasi dapat diterapkan untuk melakukan enkripsi terhadap keseluruhan blok *Plaintext*. Empat mode operasi yang lazim diterapkan pada sistem blok chiper adalah :

a. *Electronik Code Book* (ECB) Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen menjadi blok cipherteks C_i .

b. *Cipher Blok Chaining* (CBC), Mode ini menerapkan mekanisme umpan-balik (*feedback*)pada sebuah blok, yang dalam hal ini hasil enkripsi sebelumnya diumpan-balikkan ke dalam enkripsi blok yang *current*. Caranya, *blok plainteks* yang current di-XOR-kan terlebih dahulu dengan blok cipherteks hasil enkripsi sebelumnya, selanjutnya hasil peng-XORan ini masuk ke dalam fungsi enkripsi. Dengan mode CBC, setiap *blok cipherteks* bergantung tidak hanya pada blok plainteksnya tetapi juga pada seluruh blok plainteks sebelumnya. Dekripsi dilakukan dengan memasukkan *blok* cipherteks-nya yang current ke fungsi sebelumnya. Dalam hal ini, blok cipherteks sebelumnya berfungsi sebagai umpan-maju (*feedforward*) pada akhir proses dekripsi.

c. *Cipher-Feedback* (CFB) Jika mode CFB yang diterapkan untuk transmisi data, maka enkripsi tidak dapat dilakukan bila blok plainteks yang diterima belum lengkap. Secara umum, CFB p-bit mengengkripsi plainteks sebanyak p bit setiap kalinya, yang dalam hal ini mana $p \leq n$ (n = ukuran blok). Dengan kata lain, CFB mengenkripsi *cipher blok* seperti pada *cipher* aliran.

d. *Output-Feedback* (OFB) Mode OFB ini mirip dengan mode CFB, kecuali p-bit dari hasil enkripsi terhadap antrian disalin menjadi elemen posisi paling kanan di antrian. Dekripsi dilakukan sebagai kebalikan dari proses enkripsi (Mukti Qamal ; 2014 : 18 - 19)

II.5.2. Kategori Kriptografi

Kriptografi dikategorikan menjadi dua yaitu kriptografi klasik dan kriptografi modern.

1. Kriptografi klasik

   Kriptografi klasik adalah kriptografi yang berbasis karakter (enkripsi dan dekripsi dilakukan pada setiap karakter).

   Kriptografi klasik dibagi menjadi dua yaitu cipher transposisi yang mengubah susunan huruf-huruf didalam pesan dancipher substitusi yang mengganti setiap huruf atau kelompok huruf dengan sebuah huruf atau kelompok huruf lain, diantara sekian banyak algoritma kriptografi cipher subtitusi dan cipher transposisi, ada yang disebut dengan Affine cipher dan Vigenere cipher[3]. Modifikasi Affine cipher yang diperkuat dengan Vigenere cipher memberikan penyandian baru dengan cara menggabungkan dua metode yaitu Affine cipher dengan Vigenere cipher, sehingga pesan atau informasi lebih sulit untuk dipecahkan oleh kriptanalisis dibandingkan dengan penyandian yang menggunakan satu metode, misalkan hanya menggunakan Affine cipher atau Vigenere cipher saja

2. Kriptografi modern

   Kriptografi modern adalah kriptografi yang beroperasi dalam mode bit (dinyatakan dalam 0 dan 1) (Juliadi ; 2013 : 87)

II.6. 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.7. Pengertian NetBeans

NetBeans merupakan salah satu proyek *open source* yang disponsori oleh *Sun Microsystem*. Proyek ini bediri pada tahun 2000 dan telah menghasilkan 2 produk, yaitu *NetBeans* IDE dan *NetBeans Platform*. *NetBeans* IDE merupakan produk yang digunakan untuk melakukan pemrograman baik menulis kode, meng-*compile*, mencari kesalahan dan mendistribusikan program. Sedangkan *NetBeans Platform* adalah sebuah modul yang merupakan kerangka awal / pondasi dalam bangun aplikasi desktop yang besar.

NetBeans juga menyediakan paket yang lengkap dalam pemrograman dari pemrograman standar (aplikasi desktop), pemrograman enterprise, dan pemrograman perangkat mobile. Saat ini *NetBeans* telah mencapai versi 6.8 (Wahana Komputer ; 2010 : 15).

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

Menurut Sri Dharwiyanti (2013) *Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

1. *Use Case* Diagram

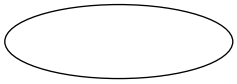
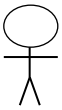
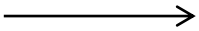
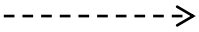
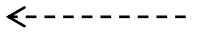
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

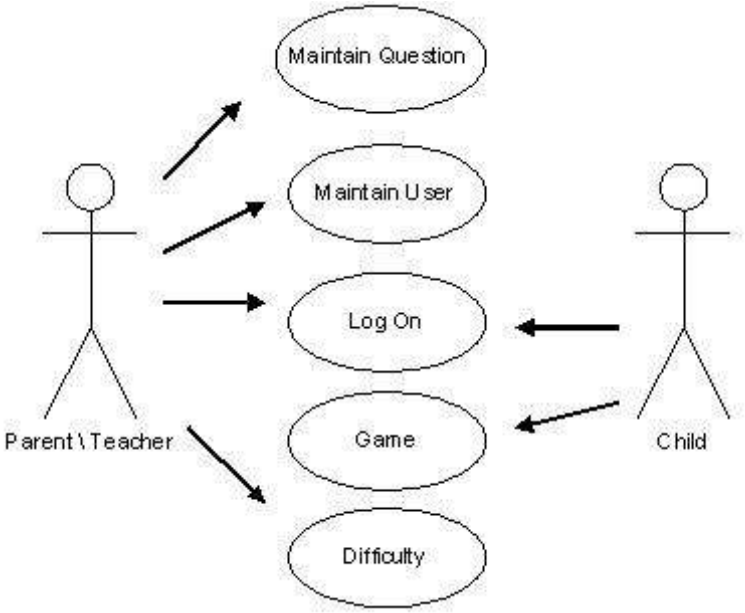
Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu

merupakan spesialisasi dari yang lain.

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan 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 mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(*Sumber : Sri Dharwiyanti ; 2013 : 5*)



Gambar II.2. *Usecase* Diagram

(Sumber : Sri Dharwiyanti ; 2013 : 5)

2. *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

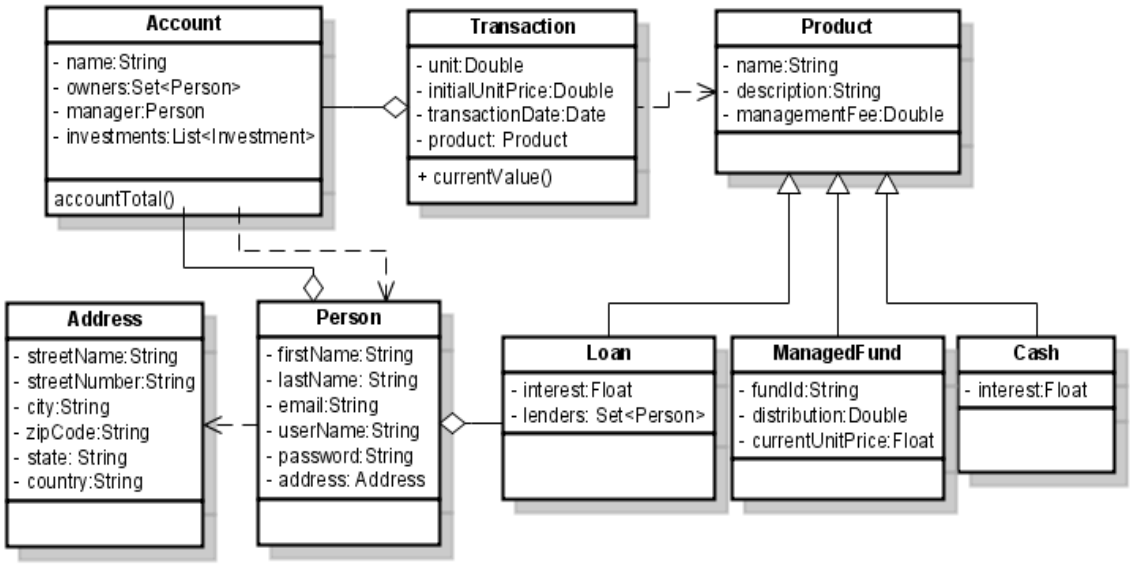
Class diagram menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Tabel II.2. *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)



Gambar II.3. *Class* Diagram

(Sumber : Sri Dharwiyanti ; 2013 : 6)

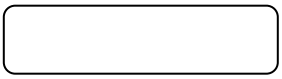
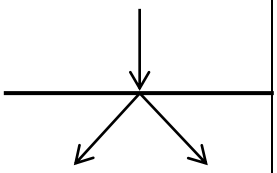
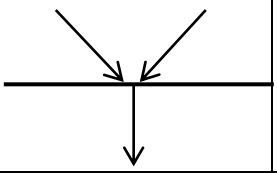
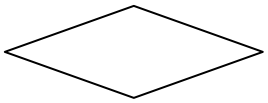
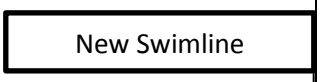
3. *Activity* Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya

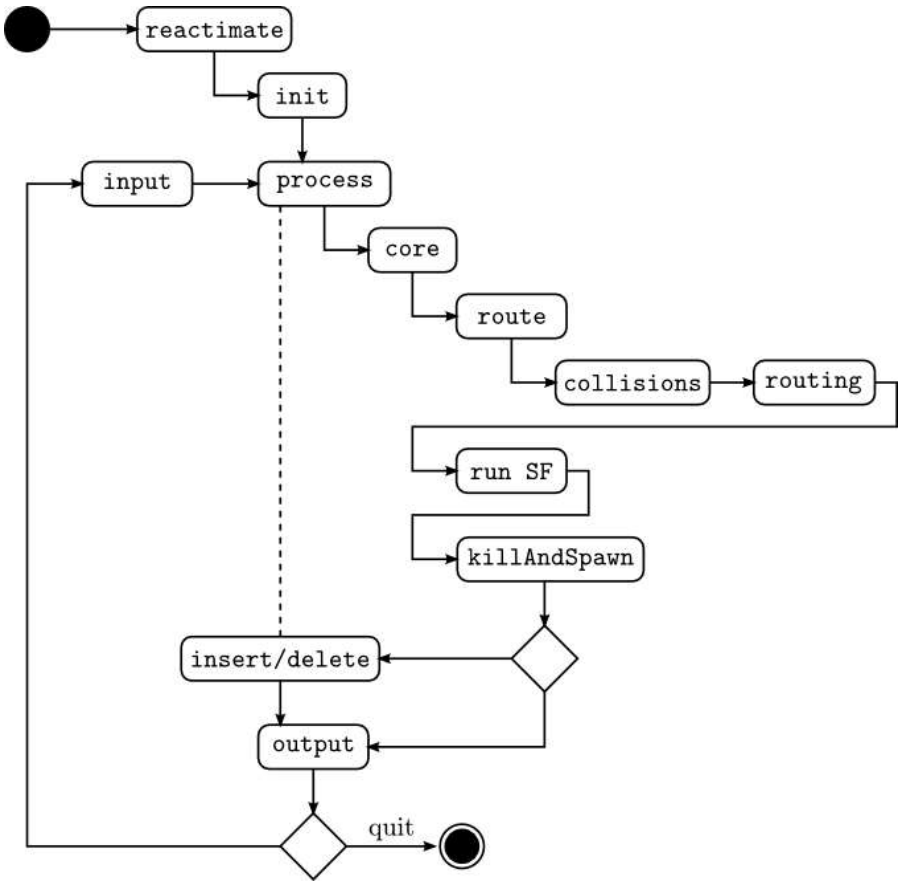
state sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

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

Tabel II.3. Simbol *Activity Diagram*

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

(***Sumber : Sri Dharwiyanti ; 2013 :***)



Gambar II.4. *Activity* Diagram

(Sumber : Sri Dharwiyanti ; 2013 : 8)

4. *Sequence* Diagram

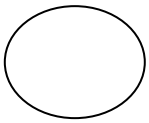
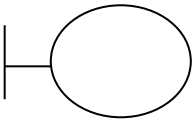
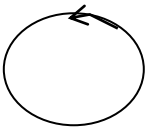
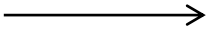
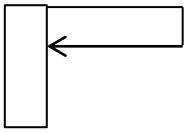
Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah

yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali

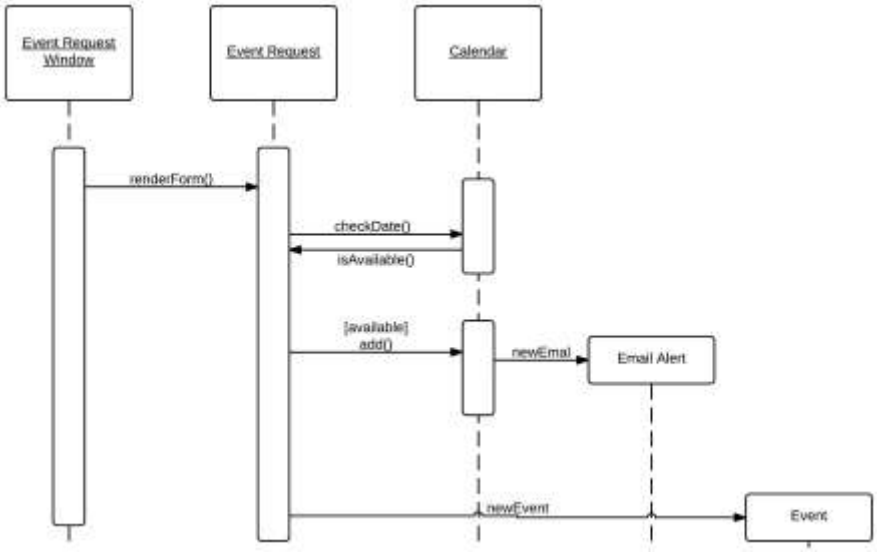
dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message.

Tabel II.4. Simbol *Sequence Diagram*

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

(***Sumber : Sri Dharwiyanti ; 2013 : 9***)



Gambar II.5. *Sequence* Diagram

(Sumber : Sri Dharwiyanti ; 2013 : 9)