BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Perancangan

Menurut jurnal Ahmad Afandi dikutip dari KBBI (Kamus Besar Bahasa Indonesia) perancangan adalah menata atau mengatur sesuatu yang diinginkan. Sementara perancangan sistem menentukan bagaimana suatu sistem akan menyelesaikan apa yang harus diselesaikan, tahap ini menyangkut mengkofigurasikan dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem sehingga setelah instalasi dari suatu sistem akan benar-benar memuaskan rancang bangunan yang telah ditetapkan pada akhir analisis sistem.

Sehingga dari pengertian di atas perancangan adalah mengatur, menata menetukan sesuatu itu menyelesaikan apa yang harus diselesaikannya (Sumber : Ahmad Afandi; 2015:2).

II.2. Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi sutau pake disebut sebagai suatu paket atau deretan aplikasi (*application suite*). Contohnya adalah Microsoft Office dan OpenOffice.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat

dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah.

Jenis-jenis *Software* Aplikasi :

1. *Software* aplikasi hiburan, contohnya yaitu winamp untuk mendengarkan musik, games dan sebagainya untuk hiburan.
2. *Softwar*e aplikasi pendidikan yaitu *software* digunakan untuk mempelajari atau mereferensikan tentang pendidikan atau pengetahuan.
3. *Software* aplikasi bisnis yaitu *software* yang digunakan untuk aplikasi bisnis
4. *Software* aplikasi khusus.
5. *Software* aplikasi untuk produtivitas kerja (Sumber : Jurnal Dahlan Abdullah, Cut Ita Erliana; 2012:3).

II.3. SMS (*Short Message Service*)

II.3.1. Definisi SMS

Short Message Service (SMS) (Talukder, 2005.) merupakan sebuah layanan yang banyak diaplikasikan pada sistem komunikasi tanpa kabel, memungkinkan dilakukannya pengiriman pesan dalam bentuk teks. SMS didukung oleh GSM (*Global System For Mobile Communication*), TDMA (*Time Division Multiple Access*), CDMA (*Code Division Multiple Access*) yang berbasis pada telepon seluler yang saat ini banyak digunakan. SMS (*Short Message Service*) adalah merupakan salah satu layanan pesan teks yang dikembangkan dan distandarisasi oleh suatu badan yang bernama ETSI (*European Telecommunication Standards Institute*) sebagian dari pengembangan GSM *(Global System for Mobile Communication) Phase* 2, yang terdapat pada dokumentasi GSM 03.40 dan GSM 03.38. Fitur SMS ini memungkinkan perangkat

Stasiun Seluler Digital (*Digital Cellular Terminal*, seperti Ponsel) untuk dapat mengirim dan menerima pesan-pesan teks dengan panjang sampai dengan 160 karakter melalui jaringan GSM.

SMS dapat dikirimkan ke perangkat stasiun seluler *digital* lainnya hanya dalam beberapa detik selama berada pada jangkauan pelayanan GSM. Lebih dari sekedar pengiriman pesan biasa, layanan SMS memberikan garansi SMS akan sampai pada tujuan meskipun perangkat yang dituju sedang tidak aktif yang dapat disebabkan karena sedang dalam kondisi mati atau berada di luar jangkauan layanan GSM. Dengan adanya *feature* seperti ini maka layanan SMS juga cocok untuk dikembangkan sebagai aplikasi-aplikasi seperti: *pager*, *e-mail*, dan notifikasi *voice mail*, serta layanan pesan banyak pemakai (*multiple user*). Namun pengembangan aplikasi tersebut masih bergantung pada tingkat layanan yang disediakan oleh operator jaringan (Sumber : Yudi Wiharto; 2011:2).

II.3.2. Karakteristik SMS

Karakteristik utama SMS adalah SMS merupakan sebuah sistem pengiriman data dalam paket yang bersifat *out-of-band* dengan *bandwith* kecil. Dengan karakteristik ini, pengiriman suatu *burst* data yang sangat pendek dapat dilakukan dengan efisiensi yang sangat tinggi (Sumber : Yudi Wiharto; 2011:2).

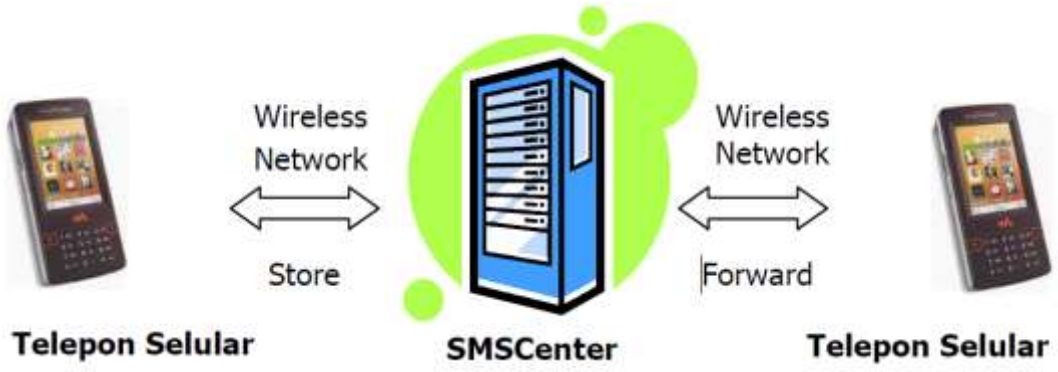
II.3.3. Keuntungan SMS

Pada tingkat minimum keuntungan yang dapat diberikan oleh SMS bagi pemakai meliputi pengiriman notifikasi dan peringatan (*alert*), penyampaian pesan SMS yang terjamin, handal, mekanisme komunikasi dengan biaya rendah, kemampuan untuk menyaring pesan SMS dan menanggapi panggilan secara selektif sehingga meningkatnya produktifitas *customer*. Untuk fungsionalitas yang lebih canggih, SMS memberikan beberapa keuntungan tambahan bagi *user*

yaitu pengiriman pesan SMS ke beberapa *user* sekaligus dalam waktu yang bersamaan, kemampuan menerima informasi yang beragam, dan integrasi dengan aplikasi lain yang berbasis internet dan data (Sumber : Yudi Wiharto; 2011:2).

II.3.4. Cara Kerja SMS

Dalam sistem SMS, mekanisme utama yang dilakukan dalam suatu sistem adalah melakukan pengiriman *short message* dari satu terminal *customer* ke terminal yang lain. Hal ini dapat dilakukan berkat adanya sebuah entitas dalam sistem SMS yang bernama *Short Message Service Center* (SMSC), disebut juga *Message Center* (MC). Pada saat pesan SMS dikirim dari *handphone* (*mobile orginated*) pesan tersebut tidak langsung dikirim ke *handphone* tujuan (*mobile terminated*), akan tetapi terlebih dahulu ke SMSC, baru kemudian pesan tersebut dikirimkan ke *handphone* tujuan. SMSC merupakan sebuah perangkat yang melakukan tugas *store* and *forward* trafik *short message*. Di dalamnya termasuk penentuan atau pencarian rute tujuan akhir dari *short message*. Sebuah SMSC biasanya didesain untuk dapat menangani *short message* dari berbagai sumber seperti *Voice Mail System* (VMS), *Web-based messaging*, *Email Integration*, *External Short Message Entities* (ESME), dan lain-lain (Sumber : Yudi Wiharto; 2011:2).



Gambar II.1. Skema Cara Kerja SMS

(Sumber : Yudi Wiharto; 2011:3)

II.4. Keamanan Data

Keamanan data pada lalu lintas jaringan adalah suatu hal yang sangan diinginkan semua orang untuk menjaga *privacy*. Supaya data yang dikirim aman dari orang yang tidak bertanggung jawab dengan menyembunyikan data memakai algoritma kriptografi (Sumber : Dony Ariyus; 2006:9).

II.4.1. Kriptografi

Kriptografi berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua kripto dan graphia, kripto berarti *secret* (rahasia) dan graphia berarti *writing* (tulisan). Menurut terminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ketempat yang lain (Sumber : Dony Ariyus; 2006:9).

II.4.2. Sejarah Kriptografi

Kriptografi mempunyai sejarah yang sangat menarik dan panjang. Kriptografi sudah digunakan 4000 tahun yang lalu yang diperkenalkan oleh orang-orang mesir untuk mengirim pesan ke pasukan militer yang berada di lapangan dan supaya pesan tersebut tidak terbaca oleh pihak musuh walaupun kurir pembawa pesan tersebut tertangkap oleh musuh.

Pada zaman romawi kuno dikisahkan pada suatu saat, ketika Julius Caesar ingin mengirimkan satu pesan rahasi kepada seorang Jenderal di medan perang. Pesan tersebut harus dikirimkan melalui seorang kurir, tetapi karena pesan tersebut mengandung rahasia, Julius Caesar tidak ingin pesan tersebut terbuka di tengah jalan. Di sini Julius Caesar tidak ingin pesan tersebut terbuka di tengah jalan. Di sini Julius Caesar memikirkan bagaimana mengatasinya yaitu dengan mengacak pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapapun

kecuali hanya dapat dipahami oleh Jenderal saja. Tentu sang Jenderal telah diberi tahu sebelumnya bagaimana cara membaca pesan yang teracak tersebut, karena telah mengetahui kuncinya. Yang dilakukan Julius Caesar adalah mengganti semua susunan alfaber dari a, b, c & yaitu a menjadi d, b menjadi e, c menjadi f dan seterusnya. Sehingga kalau Julius menuliskan kata " saya sekarang & vhndudqi & membacanya (Sumber : Dony Ariyus; 2006:9).

Dari ilustrasi tersebut, beberapa istilah *Cryptography* dipergunakan untuk menandai aktifitas-aktifitas rahasia dalam mengirimkan pesan. Apa yang dilakukan Julius Caesar dengan cara mengacak pesannya, kita sebut sebagai *encryption* dan pada saat Sang Jendral merapikan pesan yang teracak itu, kita sebut dengan *decryption*. Pesan awal yang belum diacak dan yang telah dirapikan, kita sebut *plaintext* sedangkan pesan yang telah diacak, kita sebut *ciphertext*. Huruf-huruf dengan bentuk tegak akan mempunyai lebar huruf yang lebih kecil dibandingkan dengan huruf-huruf yang melintang, sehingga dengan jumlah huruf yang sama, huruf yang membentjk melintang akan memakan banyak tempat. Spasi antar huruf juga terlihat bervariasi pada huruf yang melintang daripada huruf tegak (Sumber : Dony Ariyus; 2006:10).

II.4.3. Konsep Dasar Kritopgrafi

Kriptografi adalah suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim ke penerima tanpa mengalami gangguan dari pihak ketiga. Menurut Bruce Scheiner dalam bukunya "*Applied Cryptography*", kriptografi adalah ilmu pengetahuan dan seni menjaga pesan tetap aman (*secure*).

Konsep kriptografi sendiri telah lama digunakan oleh manusia misalnya pada peradaban Mesir dan Romawi walau masih sangat sederhana. Prinsip-prinsip yang mendasari kriptografi yakni:

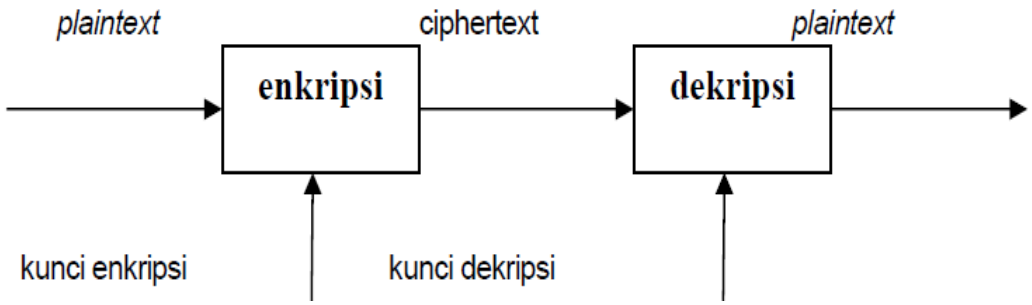
1. *Confidelity* (kerahasiaan) yaitu layanan agar isi pesan yang dikirimkan tetap rahasia dan tidak diketahui oleh pihak lain (kecuali pihak pengirim, pihak penerima / pihak-pihak memiliki ijin). Umumnya hal ini dilakukan dengan cara membuat suatu algoritma matematis yang mampu mengubah data hingga menjadi sulit untuk dibaca dan dipahami.
2. *Data integrity* (keutuhan data) yaitu layanan yang mampu mengenali/mendeteksi adanya manipulasi (penghapusan, pengubahan atau penambahan) data yang tidak sah (oleh pihak lain).
3. *Authentication* (keotentikan) yaitu layanan yang berhubungan dengan identifikasi. Baik otentikasi pihak-pihak yang terlibat dalam pengiriman data maupun otentikasi keaslian data/informasi.
4. *Non-repudiation* (anti-penyangkalan) yaitu layanan yang dapat mencegah suatu pihak untuk menyangkal aksi yang dilakukan sebelumnya (menyangkal bahwa pesan tersebut berasal dirinya).

Istilah-istilah yang digunakan dalam bidang kriptografi :

1. *Plaintext* (M) adalah pesan yang hendak dikirimkan (berisi data asli).
2. *Ciphertext* (C) adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
3. Enkripsi (fungsi E) adalah proses pengubahan *plaintext* menjadi *ciphertext*.
4. Dekripsi (fungsi D) adalah kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga berupa data awal/asli.

Kriptografi itu sendiri terdiri dari dua proses utama yakni proses enkripsi dan proses dekripsi. Seperti yang telah dijelaskan di atas, proses enkripsi mengubah *plaintext* menjadi *ciphertext* (dengan menggunakan kunci tertentu) sehingga isi informasi pada pesan tersebut sukar dimengerti (Sumber : Muhammad Fairuzabadi; 2010:66).

.



Gambar II.2. Diagram Proses Enkripsi Dan Dekripsi
(Sumber : Muhammad Fairuzabadi; 2010:67)

II.4.3. Algoritama Kriptografi

Algoritama ditinjau dari asal usul kata, kata algoritma mempunyai sejarah yang menarik, kata ini muncil didalam kamus Webster sampai akhir tahun 1957 hanya menemukan kata algorim yang mempunyai arti proses perhitungan dengan bahasa Arab. Algoritma berasal dari nama penulis buku Arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca oleh orang barat menjadi *algorism*). Kata *algorism* lambat laun berubah menjadi *algorithm*.

Defenisi Terminologinya Algoritma adalah urutan langkah-langkah logis untuk penyelesaian masalah uang disusun sedara sistimatis. Algoritma kriptografi merupakan langkah-langkah logis bagaiman menyembunyikankan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

Algoritma kriptogragi terdiri dari tiga fungsi dasar yaitu:

1. Enkripsi

Enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan terjaga rahasiannya, pesan asli disebur plaintext yang dirubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan *chiper* atau kode.

Sama halnya dengan kita tidak mengerti akan sebuah kata, maka kita akan melihatnya di dalam kamus atau daftar istilah-istilah. Beda halnya dengan enkripsi, untuk merubah plaintext ke bentuk ciphertext kita menggunakan algoritma yang dapat mengkodekan data yang kita inginkan.

2. Deskripsi

Dekripsi merupakan kebalikan dari enkripsi, pesan yang telah dikembalikan kebentuk asalnya (*Plaintext*) disebut dengan deskripsi pesan. Algoritma yang digunakan untuk dekrisi tertu berbeda dengan yang digunakan untuk enkripsi.

3. Kunci

kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi, kunci terbagi jadi dua bagian kunci pribadi (*private key*) dan kunci umum (*public key*).

Keamanan dari algoritma kriptografi tergantung dari bagaimana suatu algoritma itu bekerja, maka algoritma semacam ini disebut dengan algoritma terbatas. Algoritma terbatas merupakan suatu algoritma yang dipakai sekelompok orang untuk merahasiakan pesan yang dikirimnya. Jika salah satu dari anggota kelompok itu keluar dari kelompoknya maka, algoritma yang dipakai diganti dengan yang baru, jika tidak hal ini bisa menjadi masalah dikemudian hari.

Keamanan dari kriptografi modern hanya dengan merahasiakan kunci yang dimiliki orang lain, tanpa harus merahasiakan algoritma itu sendiri. Kunci berfungsi sama seperti password. Jika keseluruhan dari keamanan algoritma tergantung pada kunci yang dipakai maka, algoritma ini bisa dipublikasikan dan dianalisi oleh orang lain. Jika algoritma yang telah dipublikasikan bisa dipecahkan dalam waktu singkat oleh orang lain maka, algoritma tersebut belum aman untuk digunakan. Pada pembahasan berikutnya akan dijelaskan berbagai macam algoritma kriptografi yang pernah ada (Sumber : Dony Ariyus; 2006:13).

II.4.4. Macam Macam Algoritama Kriptografi

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan dari kunci yang dipakainya:

1. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya).
2. Algoritma Asimetri (menggunakan satu kunci untuk enkripsi dan dekripsinya)
3. *Hash Function*.

II.4.4.1. Algoritma Simetri

Algoritma ini juga sering disebut dengan algorima klasik, karena pakai kunci yang sama untuk kegiatan enkripsi dan dekripsinya. Algoritma ini sudah ada lebih dari 4000 tahun yang lalu. Mengirim pesan dengan menggunakan algoritma ini, sipenerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsi pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci, jika kunci tersebut diketahui oleh orang lain maka, orang tersebut bisa melakukan enkripsi dan dekripsi terhadap pesan tersebut. Algoritma yang memakai kunci simetri diantaranya adalah:

1. Data Encryption Standard (DES)
2. RC2, RC4, RC5, RC6.
3. International Data Encryption Algorithm (IDEA).
4. Advanced Encryption Standard (AES).
5. One Time Pad (OTP).
6. A5.
7. Dan lain sebagainya.

II.4.4.2. Algoritma Asimetri

Algoritma asimetri sering juga disebut dengan algoritma kunci public, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsinya berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian:

1. Kunci umum (*public key*): Kunci yang boleh semua orang tahu (dipublikasikan)
2. Kunci pribadi (*private key*): Kunci yang dirahasiakan (hanya boleh diketahui oleh satu orang).

Kunci-kunci tersebut saling berhubungan satu dengan yang lainnya. Dengan kunci public orang dapat mengenkripsi pesan tapi tidak bisa mendekripsinya, hanya orang yang memiliki kunci pribadi yang dapat mendekripsi pesan tersebut. Algoritma asimetris bisa melakukan pengiriman pesan lebih aman dari pada algoritma simetri. Contoh: Bob mengirim pesan ke Alice menggunakan algoritma asimetris, hal yang harus dilakukan adalah:

1. Bob memberitahukan kunci publikasinya ke Alice.
2. Alice mengenkripsi pesan dengan menggunakan kunci public Bob.
3. Bob mendekripsi pesan dari alice dengan kunci pribadinya.
4. Dan begitu juga sebaliknya jika Bob ingi mengirim pesan ke Alice.

Algoritma yang memakai kunci publik diantaranya adalah:

1. *Digital Signature Algorithm* (DSA).
2. RSA.
3. *Diffie-Hellman* (DH).

II.4.4.3. *Hash Function* (Fungsi Hash)

Fungsi hash sering disebut dengan fungsi hash satu arah (*one-way function*), messege digest, fingerprint, fungsi kompresi dan *Messege Authentication Code* (MAC), hal ini merupakan suatu fungsi matematika yang mengambil input panjang veriable dan mengubahnya ke dalam

urutan biner dengan panjang yang tetap. Fungsi hash biasanya diperlakukan bila ingin membuat sidik jari dari suatu pesan. Sidik jari pada pesan merupakan suatu tanda yang menandakan bahwa pesan tersebut benar-benar dari orang yang diinginkan (Sumber : Dony Ariyus; 2006:14).

II.4.5. Kriptografi Klasik

Kriptografi klasik merupakan suatu algoritma yang menggunakan satu kunci untuk mengamankan data, teknik ini sudah digunakan beberapa abad yang lalu. Dua teknik dasar yang biasa digunakan pada algoritma jenis ini, diantaranya adalah:

1. Teknik Subtitusi: Penggantian setiap karakter plaintext dengan karakter lain.
2. Teknik Transposisi (Permutasi): Teknik ini menggunakan permutasi karakter.

II.4.6. Kriptografi Modern

Kriptografi modern merupakan suatu algoritma yang digunakan pada saat sekarang ini, yang mana kriptografi modern mempunyai kerumitan yang sangat komplek, karena dalam pengoperasiannya menggunakan komputer (Sumber : Dony Ariyus; 2006:16).

II.5. *Vigenere Cipher*

Vigenere Chiper termasuk dalam cipher abjad majemuk (*polyalphabetic substitution Chiper*) yang dipublikasikan oleh diplomat (sekaligus seorang kriptologis) Perancis, Blaise de Vigenere pada abad 16 (tahun 1586). *Vigenere Chiper* adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi *Caesar* berdasarkan huruf-huruf pada kata kunci. *Vigenere Chiper* menggunakan tabel seperti pada tabel II.1, *Vigenere Cipher* dengan angka. Dalam melakukan enkripsi. Teknik dari substitusi *vigenere cipher* bisa dilakukan dengan dua cara:

1. Angka

2. Huruf

II.5.1. *Vigenere Cipher* Angka

Vigenere cipher menggunakan angka hampir mirip dengan menggunakan metode caesar, hanya saja caesar menggunakan geseran dengan satu angka yang ditentukan sedangkan vigenere cipher menggunakan *key* (lebih dari satu angka/huruf). Berikut *vigenere cipher* dengan angka :

Tabel II.1. *Index* Alfabet *Vigenere Cipher*

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

(Sumber : Putu H. Arjana et al; 2012:165)

Jika ditukar dengan angka, maka kunci dengan huruf " HARI" K = (7, 0, 17, 8) dan *plaintext* nya " SAYA HARIYANTO " akan menjadi P = (18, 0, 24, 0, 7, 0, 17, 8, 24, 0, 13, 19, 14).

Tabel II.2. Vigenere Cipher Dengan Angka

S	**A**	**Y**	**A**	**H**	**A**	**R**	**I**	**Y**	**A**	**N**	**T**	**O**
18	0	24	0	7	0	17	8	24	0	13	19	14
7	0	17	8	7	0	17	8	7	0	17	8	7
25	0	16	8	14	0	9	16	6	0	5	2	21

(Sumber : Putu H. Arjana et al; 2012:165)

Jadi *chipertext* yang dihasilkan 25, 0, 16, 8, 14, 0, 9, 16, 6, 0, 5, 2, 21, *Chipertext* yang dihasilkan dengan huruf menjadi "ZAQIOAJQGAFCV". Untuk melakukan deskripsi, bisa juga digunakan modulo 26. (Sumber : Putu H. Arjana et al; 2012:165).

II.5.2. *Vigenere Cipher* Huruf

Tabel II.3, *Vigenere Cipher dengan huruf* berisi alfabet yang dituliskan dalam 26 baris, masing-masing baris digeser ke kiri dari baris sebelumnya membentuk ke-26 kemungkinan sandi caesar setiap huruf disediakan dengan menggunakan baris yang berbeda-beda sesuai kunci yang diulang.

Tabel II.3. *Vigenere Cipher* Dengan Huruf

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

(Sumber : Putu H. Arjana et al; 2012:165)

Plantext : SAYA HARIYANTO

Kunci : HARI

Dari *plaintext* dengan kata kunci di tabel didapatkan *chipertext* sebagai berikut:

Chipertext: ZAPIOAIQFAEBW

Proses dekripsi, dilakukan dengan mencari huruf *chipertext* pada baris *plaintext* dari kata kunci. Dari contoh tabel, maka dapat disimpulkan bahwa

rumus dari enkripsi dan dekripsi data *vigenere chiper* adalah:

Enkripsi: Ci = (Pi + Ki) mod 26

Dekripsi: Pi = (Ci – Ki) mod 26; untuk Ci > = Ki

Pi = (Ci+ 26 – Ki) mod 26; untuk Ci < = Ki

Keterangan:

C = *Chiphertext*

P = *Plaintext*

K = Kunci

(Sumber : Putu H. Arjana et al; 2012:165).

II.6. Metode *Vernam Cipher* (*One Time Pad* (OTP))

Kriptografi bagi kebanyakan orang adalah sesuatu yang sangat sulit dan kita sebagai pemula cenderung malas untuk mempelajarinya. Namun ada sebuah metode kriptografi yang agak mudah untuk dipelajari dan para ahlipun telah menyatakan bahwa metode ini merupakan metode kriptografi yang cukup aman untuk digunakan. Metode tersebut biasa dikenal dengan nama *One Time Pad* (OTP) atau yang lebih dikenal dengan sebutan *Vernam Cipher*. *Vernam Cipher* diciptakan oleh Mayor J. Maugborne dan G. Vernam pada tahun 1917.

Algoritma *One Time Pad* (OTP) merupakan algoritma berjenis *symetric key* yang artinya bahwa kunci yang digunakan untuk melakukan enkripsi dan dekripsi merupakan kunci yang sama. Dalam proses enkripsi, algoritma ini menggunakan cara *stream cipher* yang berasal dari hasil XOR antara bit *plaintext* dan bit *key*. Pada metode ini *plaintext* diubah kedalam kode ASCII

dan kemudian dikenakan operasi XOR terhadap kunci yang sudah diubah ke dalam kode ASCII (Sumber : M. Sholeh, J.V. Hamokwarong; 2011:10).

Secara teoritis, teknik *one-time pad* merupakan teknik enkripsi yang sempurna (*perfect encryption*) asalkan proses pembuatan kunci benar acak.

1. Proses enkripsi *one-time pad*

10010111001011101001 naskah asli

01001110001101001101 kunci

11011001000110100100 naskah acak

Dengan *one-time pad*, operasi *exclusive or* (XOR) dilakukan pada naskah asli dan kunci secara *bitwise* seperti proses enkripsi di atas. Operasi XOR menghasilkan 0 jika argumen sama (0 dengan 0 atau 1 dengan 1) dan menghasilkan 1 jika argumen berbeda (0 dengan 1 atau 1 dengan 0). Jadi bit pertama naskah asli (1) dengan bit pertama kunci (0) menghasilkan bit pertama naskah acak (1), bit kedua naskah asli (0) dengan bit kedua kunci (1) menghasilkan bit kedua naskah acak (1), bit ketiga naskah asli (0) dengan bit ketiga kunci (0) menghasilkan bit ketiga naskah acak (0), dan seterusnya.

2. Proses dekripsi *one-time pad*

11011001000110100100 naskah acak

01001110001101001101 kunci

10010111001011101001 naskah asli

Proses dekripsi sama dengan enkripsi tetapi XOR dilakukan pada naskah acak, dengan kunci yang sama (kunci dekripsi sama dengan kunci enkripsi). Setiap bit dalam kunci jika

dioperasikan terhadap bit dalam naskah asli (seperti dalam proses enkripsi) kemudian dioperasikan lagi terhadap hasil operasi pertama (seperti dalam proses dekripsi) akan mendapatkan kembali bit naskah asli. Ini adalah konsekuensi sifat aljabar operasi XOR (Sumber : Sentot Kromodimoeljo; 2009:7).

Berikut adalah contoh penggunaan XOR pada ASCII :

Diketahui *plaintext* "KITA" dan *key* yang digunakan "BISA", proses pertama yang harus dilakukan adalah mengubah *plaintext* dan *key* dalam bentuk biner.

Plaintext:

K = 01001011

I = 01001001

T = 01010111

A = 01000001

Key:

B = 01000010

I = 01001001

S = 01010011

A = 01000001

Proses enkripsi XOR:

Plaintext : 01000010 01001001 01010011 01000001 = BISA

Key : 01010011 01000001 01010100 01010101 = SATU

Ciphertext : 00010001 00001000 00000111 00010100 = DC1 BS BEL DC4

Proses dekripsi XOR:

Ciphertext : 00010001 00001000 00000111 00010100 = DC1 BS BEL DC4

Key : 01010011 01000001 01010100 01010101 = SATU

Plaintext : 01000010 01001001 01010011 01000001 = BISA

II.7. Sistem Operasi Android

II.7.1. Android

Menurut Arifianto Teguh, android adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat *mobile*, semua perangkat lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Dalam definisi lain, Android merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware,* dan aplikasi inti yang dirilis oleh Google.

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi Android, sebuah file yang ditandai dengan akhiran *.apk.* file inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. File ini diunduh oleh pengguna ke perangkat *mobile* mereka. Semua kode dijadikan satu file *.apk,* dan kemudian kita sebut sebagai sebuah aplikasi (Sumber : Ahmad; 2015:2).

Pengembang memiliki beberapa pilihan ketika membuat aplikasi yang berbasis android. Kebanyakan pengembang menggunakan eclipse yang tersedia secara bebas untuk merancang dan mengembangkan aplikasi Android. Eclipse adalah IDE yang paling populer untuk pengembangan android, karena memiliki Android plug-in yang tersedia untuk memfasilitasi pengembangan android. Selain itu eclipse juga mendapatkan dukungan langsung dari Google

untuk menjadi IDE pengembangan aplikasi Android, ini terbukti dengan adanya penambahan plugins untuk eclipse untuk membuat project android dimana *source software* langsung dari situs resminya Google. Tetapi hal di atas tidak menutup kemungkinan untuk menggunakan IDE yang lain seperti Netbeans untuk melakukan pengembangan android (Sumber : Nazruddin Safaat; 2015:3).

Aplikasi android dapat dikembangkan pada sistem operasi berikut :

1. Windows XP Vista/Seven.
2. Mac OS X (Mac OS X 10.4.8. atau lebih baru).
3. Linux.

II.7.2. Android SDK *(Software Development Kit)*

Android SDK adalah tools API *(Aplication Programming Interface)* yang diperlukan untuk mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Beberapa fitur-fitur Android yang paling penting adalah mesin Virtual Dalvik yang dioptimalkan untuk perangkat mobile, *integrated browser* berdasarkan *engine open source WebKit*, Grafis yang dioptimalkan dan didukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1.0 (Opsional akselerasi perangkat keras), kemudian SQLite untuk penyimpanan data (database). Fitur-fitur android lainnya termasuk media yang mendukung audio, video, dan gambar, juga ada fitur bluetooth, EDGE, 3G dan WiFi, dengan fitur kamera, GPS, dan kompas. Selanjutnya fitur yang juga turut disediakan adalah lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, tools untuk debugging, profil dan kinerja memori, dan plugin untuk IDE Eclipse (Sumber : Alicia Sinsuw, Xaverius Najoan; 2013:2).

II.7.3. AVD *(Android Virtual Device)*

Android Virtual Device merupakan emulator untuk menjalankan aplikasi android, yang tampilannya dapat dilihat pada gambar II.3. Setiap AVD terdiri dari sebuah profil perangkat keras yang dapat mengatur pilihan untuk menentukan fitur hardware emulator. Misalnya, menentukan apakah menggunakan perangkat kamera, apakah menggunakan keyboard QWERTY fisik atau tidak, berapa banyak memori internal, dan lain-lain. AVD juga memiliki sebuah pemetaan versi Android, maksudnya kita menentukan versi dari platform Android akan berjalan pada emulator. Pilihan lain dari AVD, misalnya menentukan skin yang kita ingin gunakan pada emulator, yang memungkinkan untuk menentukan dimensi layar, tampilan, dan sebagainya. Kita juga dapat menentukan SD Card virtual untuk digunakan dengan di emulator (Sumber : Alicia Sinsuw, Xaverius Najoan; 2013:2).

Gambar II.3. Tampilan AVD
(Sumber : Alicia Sinsuw, Xaverius Najoan; 2013:3)

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

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.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

(Sumber : Sri Dharwiyanti; 2003:2).

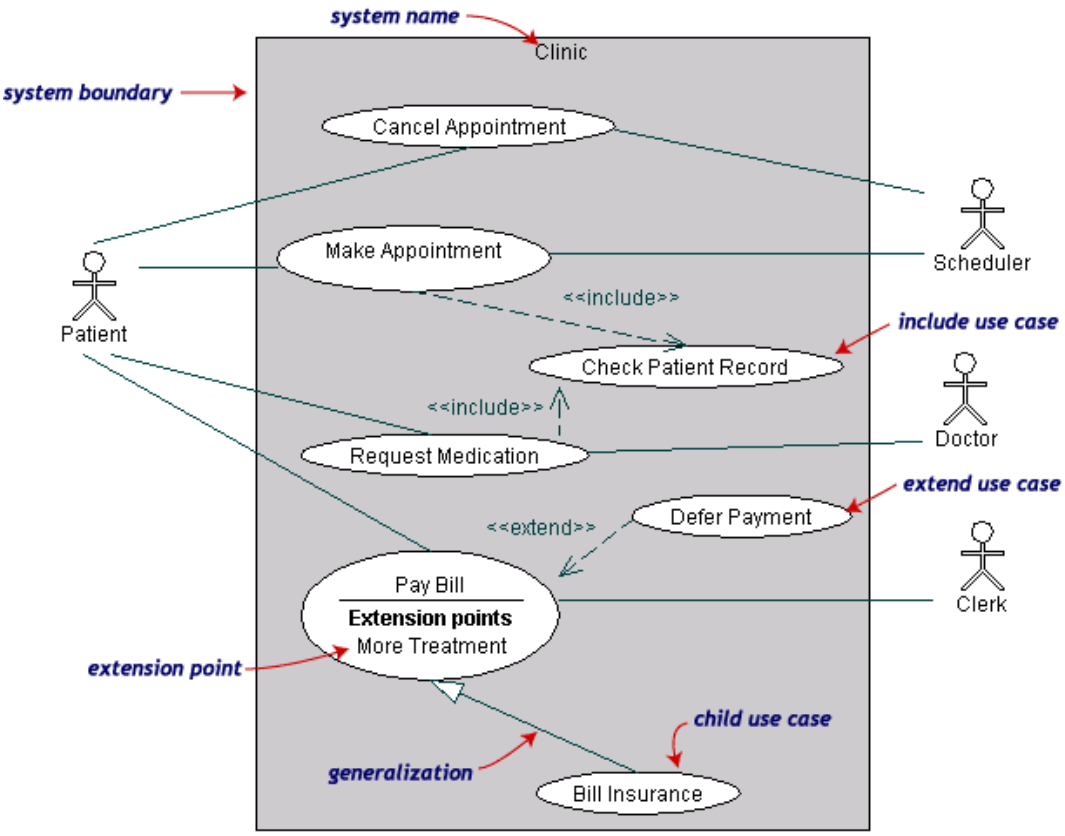
II.8.1. *Use Case Diagram*

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 sistem 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 (Sumber : Sri Dharwiyanti; 2003:4).



Gambar II.4. Contoh *Use Case Diagram*
(Sumber : Sri Dharwiyanti, 2003:5)

II.8.2. *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).

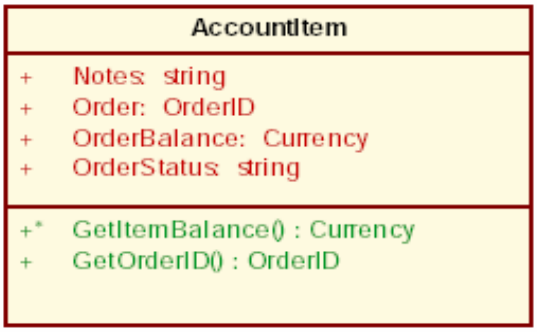
Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (*stereotype*)
2. Atribut
3. Metoda

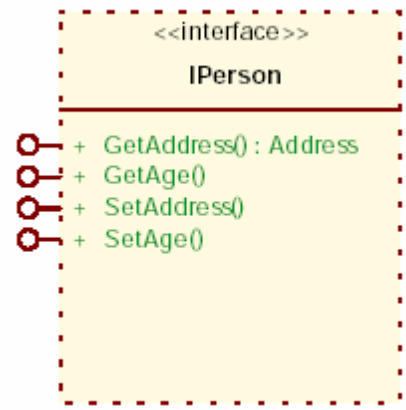
Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja



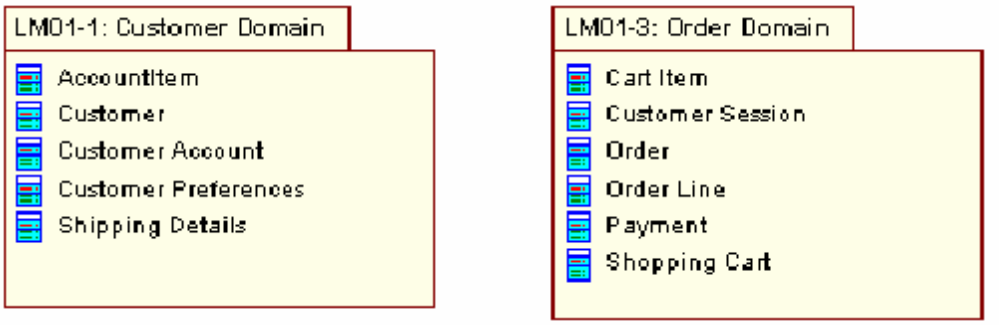
Gambar II.5. Atribut Dan Metoda Pada *Class Diagram*
(Sumber : Sri Dharwiyanti; 2003:5)

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time* (Sumber : Sri Dharwiyanti; 2003:5).



Gambar II.6. *Class Interface* Pada *Class Diagram*
(Sumber : Sri Dharwiyanti; 2003:6)

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package* (Sumber : Sri Dharwiyanti; 2003:6).

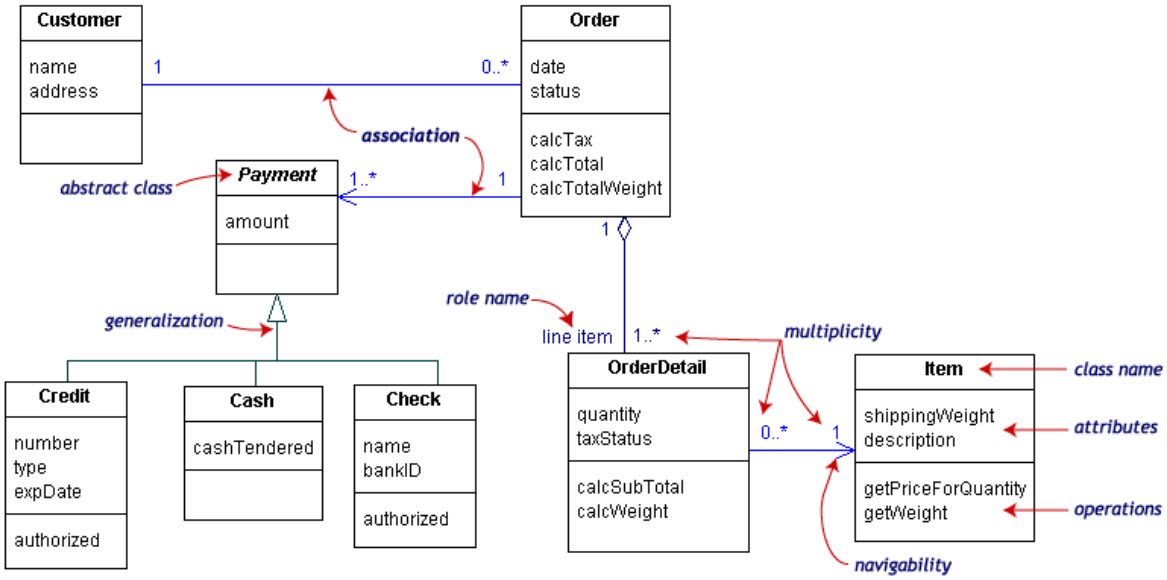


Gambar II.7. *Class* Model Dengan *Package*
(Sumber : Sri Dharwiyanti; 2003:6)

II.8.2.1. Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).

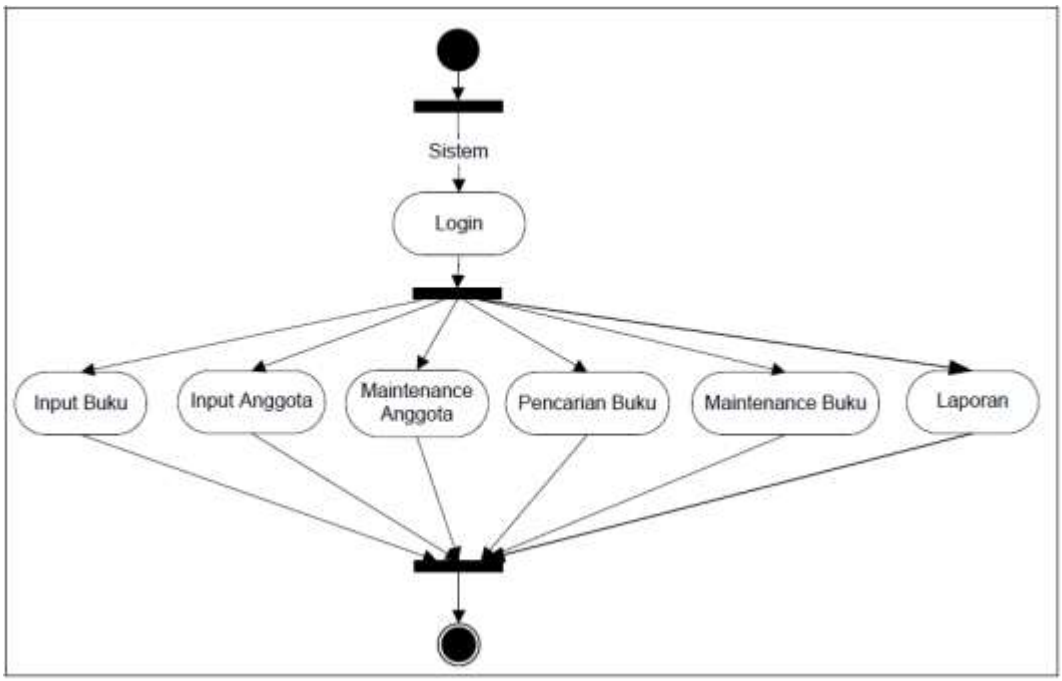
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian (Sumber : Sri Dharwiyanti; 2003:6).



Gambar II.8. Contoh *Class Diagram*
(Sumber : Sri Dharwiyanti; 2003:6)

II.8.3. *Activity Diagram*

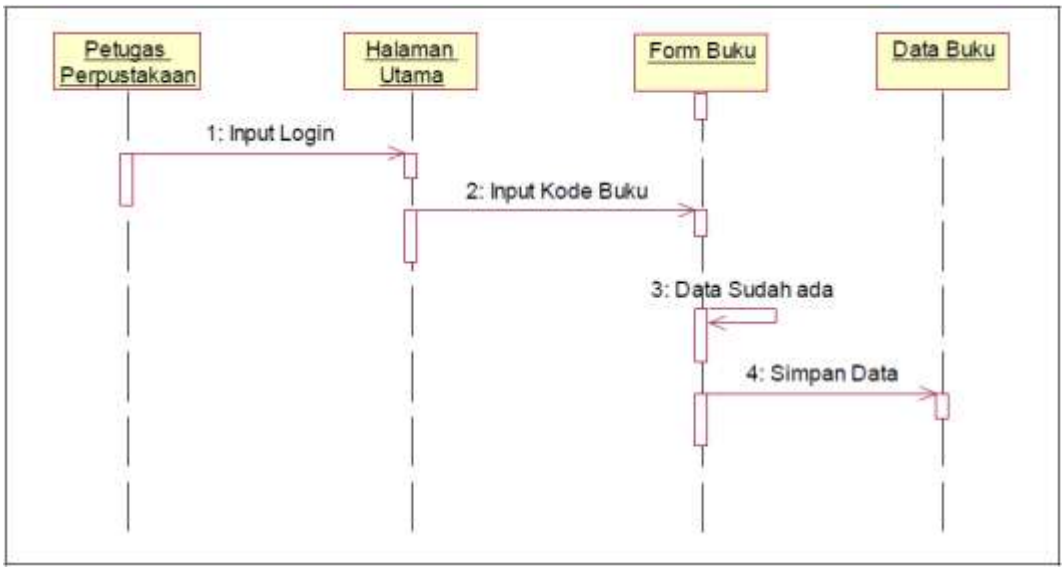
Diagram aktifitas menunjukkan aktifitas dari beberapa bagian dari struktur organisasi yang terlibat di dalam sistem. Misalkan *activity diagram* petugas menggambarkan kegiatan petugas perpustakaan yang berhubungan dengan aktivitas terhadap sistem (Sumber : Rian Fitrah; 2012:38).



Gambar II.9. *Activity Diagram* Petugas Perpustakaan
(Sumber : Rian Fitrah; 2012:39)

II.8.4. *Sequence Diagram*

Sequence menjelaskan secara detail urutan proses yang dilakukan oleh bagian-bagian yang terlibat di dalam sistem dalam sistem untuk mencapai tujuan dari *use case* interaksi terjadi antara *class*, operasi apa yang terlihat, urutan antara operasi, dan informasi yang diperlukan oleh masing-masing operasi. Misalkan *Sequence diagram* input data buku menjelaskan proses tahap-tahap input buku. Proses dimulai dari tampilan form input buku, proses input hingga ke penyimpanan data (Sumber : Rian Fitrah; 2012:43).



Gambar II.10. *Sequence Diagram* Input Buku
(Sumber : Rian Fitrah; 2012:44)