IMPLEMENTASI JAVACARD
PADA SISTEM DATABASE APOTEK

TUGAS AKHIR

Oleh

RIZKI FERDIHANTHORO
06 06 04 2891

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GENAP 2007/2008

IMPLEMENTASI JAVACARD
PADA SISTEM DATABASE APOTEK

TUGAS AKHIR

Oleh

RIZKI FERDIHANTHORO
06 06 04 2891

TUGAS AKHIR INI DIAJUKAN UNTUK MELENGKAPI
SEBAGIAN PERSYARATAN MENJADI SARJANA TEKNIK

DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNIK UNIVERSITAS INDONESIA
GENAP 2007/2008

PERNYATAAN KEASLIAN TUGAS AKHIR

Saya menyatakan dengan sesungguhnya bahwa tugas akhir dengan judul:

IMPLEMENTASI JAVACARD
PADA SISTEM DATABASE APOTEK

yang dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Pendidikan Sarjana Teknik Ekstensi Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia, sejauh yang saya ketahui bukan merupakan tiruan atau duplikasi dari tugas akhir yang sudah dipublikasikan dan atau pernah dipakai untuk mendapatkan gelar kesarjanaan di lingkungan Universitas Indonesia maupun di Perguruan Tinggi atau instansi manapun, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Depok, 14 Juli 2008

Rizki Ferdihanthoro
NPM 06 06 04 2891

PENGESAHAN

Tugas akhir dengan judul :

IMPLEMENTASI JAVACARD
PADA SISTEM DATABASE APOTEK

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Pendidikan Sarjana Teknik Ekstensi Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia. Tugas akhir ini telah diujikan pada sidang ujian tugas akhir pada tanggal 7 Juli 2008 dan dinyatakan memenuhi syarat/sah sebagai tugas akhir pada Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia.

Depok, 14 Juli 2008

Dosen Pembimbing

Prof. Dr. Ir. Harry Sudibyo S, DEA
NIP. 130 891 668

UCAPAN TERIMA KASIH

Puji dan syukur kehadirat ALLAH SWT atas berkat dan rahmat-Nya sehingga tugas akhir ini dapat diselesaikan dengan baik. Tak lupa penulis juga mengucapkan terima kasih kepada:

Prof. Dr. Ir. Harry Sudibyo S, DEA

F. Astha Ekadiyanto ST, MSc

selaku dosen pembimbing dan penasehat yang telah meluangkan waktu untuk memberi pengarahan, diskusi dan bimbingan, serta persetujuan sehingga tugas akhir ini dapat selesai dengan baik. Terima kasih pula kepada kedua orang tua dan seluruh anggota keluarga atas dukungan yang telah diberikan. Tidak lupa terima kasih kepada semua rekan-rekan yang tidak dapat disebutkan satu per satu.

Rizki Ferdihanthoro
NPM 06 06 04 2891
Departemen Teknik Elektro

Dosen Pembimbing
Prof. Dr. Ir. Harry Sudibyo S, DEA

IMPLEMENTASI JAVACARD PADA SISTEM DATABASE APOTEK

ABSTRAK

Dunia informasi pada saat ini mengalami perkembangan yang cukup pesat, terbukti dengan banyak munculnya aplikasi yang berguna dalam menangani berbagai bidang permasalahan. Pada apotek, sistem informasi hanya digunakan dalam *database* stok barang tetapi tidak dalam komunikasi dengan dokter.

Tugas akhir ini bertujuan untuk memberikan gambaran umum tentang rancang bangun dalam mengimplementasikan suatu aplikasi *java card* yang terintegrasi dengan sistem *database* apotek. *Java card* merupakan salah satu jenis dari *smart card* yang menggunakan pemrograman berbasis *java*. *Smart card* pada aplikasi ini berfungsi sebagai pengganti resep obat.

Program aplikasi ini terdiri atas tiga program aplikasi *host* yaitu aplikasi administrasi pada rumah sakit, aplikasi dokter, aplikasi pengguna pada apotek dan *applet* pada kartu. Program aplikasi ini dimodifikasi dan disimulasikan menggunakan simulator *Java Card Workstation Development Environment* (JCWDE) untuk dilakukan analisa dan pengujian. Skenario ujicoba dilakukan dengan mensimulasikan komunikasi antara *smart card* yang telah disi oleh dokter dengan aplikasi apotek yang terhubung ke sistem *database* apotek.

Hasil dari tugas akhir ini berupa rancangan suatu sistem kartu akses s*mart card* yang dapat mempercepat dan meningkatkan pelayanan pada apotek.

Kata kunci : Apotek, Implementasi, *Database, Java Card, Smart Card*

Rizki Ferdihanthoro
NPM 06 06 04 2891
Electronic Engineering Department

Counsellors
Prof. Dr. Ir. Harry Sudibyo S, DEA

**JAVA CARD IMPLEMENTATION
IN PHARMACY DATABASE SYSTEM**

ABSTRACT

Nowadays, the information age is emerging rapidly, driven by many useful applications for solving problems in almost every aspect in human life. At pharmacy, information system, currently, only used to manage inventory database but not in communication with medical doctors.

The discussion in this bachelor thesis starts with general explanation about the design, construction and ends with the discussion of its implementation in java card which is integrated with the pharmacy's database system. Java card is a smart card type using programming based on java. In this application, the smart card functions as a substitute of drug recipe.

The application program consists of three application host programs. They are the Hospital Administration application, the Medical Doctor application, the Pharmacy Consumer application and applet at card. This application program is developed and simulated using a Java Card Workstation Development Environment (JCWDE) simulator for test and analysis. The practical scenario involves interaction between smart card which had been written by doctor to pharmacy application while being connected to the pharmacy database system as back-end application.

The result of this final project is to design an access card for smart card which can improve the quality of services at pharmacy.

Keywords : Pharmacy, Implementation, Database, Java Card, Smart Card

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR SINGKATAN

AID	: Application IDentification
ATM	: Automated Teller Machine
APDU	: Aplication Protocol Data Unit
API	: Application Programming Interface
CAD	: Card Acceptance Reader
CAP	: Converted Applet
CPU	: Central Processing Unit
EEPROM	: Electrically Erasable Programmable Read Only Memory
GUI	: Graphic User Interface
IEC	: International Electronics Commitee
IC	: Integrated Circuit
ISO	: International Standardization Organization
JC	: Java Card
JCDK	: Java Card Development Kit
JCRE	: Java Card Runtime Environment
JCRMI	: Java Card Remote Method Invocation
JCWDE	: Java Card Workstation Development Environment
JCVM	: Java Card Virtual Machine
JTC	: Joint Technical Commitee
JVM	: Java Virtual Machine
J2ME	: Java 2 Mobile Environment
J2SE	: Java 2 Standard Edition
N.I	: Ne Iter
OCF	: Open Card Framework
PIN	: Personal Identification Number
PNS	: Pegawai Negeri Sipil
PT	: Perseroan Terbatas
RAM	: Read Access Memory
RF	: Radio Frequency
RFID	: Radio Frequency IDentification

RMI	: Remote Method Invocation
RMIC	: Remote Method Invocation Converter
SATSA	: Security and Trust Service Application Programing Interface
SIM	: Subscriber Identity Module
SW	: Status Word
USB	: Universal Serial Bus
VM	: Virtual Machine
WDE	: Workstation Development Environment

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG MASALAH

Smart card di Indonesia bukanlah hal baru, telah banyak perusahaan yang telah mengaplikasikan *smart card* sebagai kartu identitas.

Keterbatasan *smart card* pada saat ini adalah dalam hal memori. Di masa depan, kemungkinan besar kapasitas memori yang disediakan dalam *smart card* akan semakin bertambah besar, sehingga kemampuannya dalam hal menyimpan dan mengenkripsi data akan semakin bertambah pula. Sehingga hanya dengan membawa satu kartu akan dapat berfungsi sebagai kartu ATM, kartu kredit, kartu identitas baik dalam suatu perusahaan maupun identitas penduduk, dan masih banyak lagi fungsi yang lainnya yang dapat dimasukkan pada kartu ini.

Keunggulan dari teknologi *Java Card* adalah fleksibilitasnya yang tinggi, dimana seperti halnya bahasa pemrograman yang berorientasi objek, kita dapat memanfaatkan semua *package* yang telah ada untuk dikembangkan selama kita tahu apa yang dapat dan tidak dapat dilakukan objek tersebut. *Sun microsystems* sendiri menyertakan beberapa contoh aplikasi yang cukup bermanfaat untuk diterapkan dalam kehidupan sehari-hari, seperti aplikasi kartu pembayaran, kartu anggota, kartu akses, dan sebagainya.

Pada saat ini kita masih menggunakan kertas pada saat kita ingin membawa resep ke apotek. Sebagaimana kita tahu bahwa kertas merupakan salah satu hal yang perlu diperhatikan dalam pemanasan global. Dengan penggunaan kertas yang semakin meningkat maka akan semakin banyak pohon yang akan ditebang untuk memenuhi kebutuhan kertas di dunia. Oleh karena itu, pada tugas akhir ini dibuatlah simulasi dari smart card yang berisikan resep dokter sebagai salah satu bentuk kepedulian terhadap lingkungan hidup.

1.2 TUJUAN

Tujuan pembuatan tugas akhir yang berjudul "Implementasi Javacard pada Sistem Database Apotek" adalah:

1. Mengimplementasikan javacard pada sistem database apotek dengan menggunakan java card reference implementation.
2. Mempermudah apoteker dalam melakukan pencarian database obat pada apotek.
3. Membuat suatu sistem smart apotek yang dapat digunakan oleh banyak user dengan otoritasnya masing-masing.

1.3 PEMBATASAN MASALAH

Pembatasan masalah pada tugas akhir ini adalah sebagai berikut.

1. Tugas akhir ini merupakan pembuatan program aplikasi berbasis *smart card* yang mengimplementasikan teknologi *Java Card*, proses pengembangannya, analisa program, pengujian, dan demo program.
2. Pembahasan mencakup aplikasi apotek, aplikasi dokter dan *applet* pada kartu, sementara aplikasi *back-end* adalah program database sederhana.
3. Aplikasi ini menggunakan C-JCRE (*C language Java Card Runtime Environment*) sebagai pengganti *smart card*.

1.4 METODE PERANCANGAN

Metode perancangan menggunakan *use case* diagram, mendeskripsikan aktor-aktor dan hubunganya, sekuensial diagram menerangkan sekuensial dari program yang dibuat, kelas diagram dari applet dan aliran diagram komunikasi antara aplikasi pengguna dan applet pada kartu.

Aplikasi *smart card* sebagai dompet elektronik mengunakan *Java Card* RMI, dengan pendekatan model yang berpusat pada objek. Aplikasi dibuat menggunakan model RMI dengan membuat aplikasi *server*, remote objek dan aplikasi pengguna, dan membuat remote *reference* pada *server* remote objek.

1.5 SISTEMATIKA PENULISAN

Sistematika penulisan tugas akhir ini adalah sebagai berikut :

Bab I Pendahuluan

Membahas latar belakang masalah, tujuan, pembatasan masalah, metode perancangan, dan sistematika penulisan skripsi ini.

Bab II Teori Dasar Apotek dan Java Card

Membahas dasar dari sistem apotek yang diperlukan untuk merancang aplikasi *smart card* berbasis teknologi *Java Card*. Membahas tentang *smart card* yang berisi penjelasan tentang prinsip kerja *smart card*. Hubungan antara sistem kerja apotek dengan program yang dibuat.

Bab III Metode Perancangan Aplikasi *Smart Card* pada Sistem Database Apotek

Membahas tentang pengembangan aplikasi, *use case* diagram, diagram sekuensial aplikasi *smart card* dalam sistem database apotek.

Bab IV Pengujian dan Analisa Program

Ujicoba aplikasi sistem, analisa hasil ujicoba.

Bab V Kesimpulan

Menyimpulkan hasil dari program yang dibuat terhadap rancangan yang dihasilkan.

BAB II
TEORI DASAR APOTEK DAN JAVA CARD

2.1 APOTEK

Apotek adalah tempat dimana dilakukan pekerjaan kefarmasian dan penyaluran obat kepada masyarakat[1]. Kata ini berasal dari kata bahasa Yunani *apotheca* yang secara harfiah berarti "penyimpanan".

Tugas dan fungsi apotek adalah[2]:

1. Tempat pengabdian profesi seorang Apoteker yang telah mengucapkan sumpah jabatan.
2. Sarana Farmasi yang melakukan peracikan,pengubahan bentuk, pencampuran dan penyerahan obat atau bahan obat.
3. Sarana penyalur perbekalan farmasi yang harus menyebarkan obat yang diperlukan masyarakat secara meluas dan merata.

2.1.1 Pelayanan Apotek

Hal-hal yang membedakan apotek dengan pedagang obat eceran adalah pelayanannya. Berikut merupakan pelayanan yang terjadi pada apotek:

1. Apotek wajib dibuka untuk melayani masyarakat dari pukul 8.00 sampai 22.00.
2. Apotek wajib melayani resep dokter, dokter gigi, dan dokter hewan. Pelayanan resep sepenuhnya atas tanggung jawab Apoteker pengelola Apotek.
3. Apoteker wajib melayani resep sesuai dengan tanggung jawab dan keahlian profesinya yang dilandasi pada kepentingan masyarakat. Apoteker tidak diizinkan untuk mengganti obat generik yang ditulis di dalam resep dengan obat paten. Dalam hal pasien tidak mampu menebus obat yang tertulis di dalam resep, Apoteker wajib berkonsultasi dengan dokter untuk pemilihan obat yang lebih tepat.
4. Apoteker wajib memberikan informasi:

a. Yang berkaitan dengan penggunaan obat yang diserahkan kepada pasien.
b. Penggunaan obat secara tepat, aman, rasional atas permintaan masyarakat.

5. Apabila Apoteker menganggap bahwa dalam resep ada kekeliruan atau penulisan resep yang tidak tepat, apoteker harus memeberitahukan kepada dokter penulis resep. Bila dokter penulis resep tetap pada pendiriannya, dokter wajib membubuhkan tanda tangan yang lazim di atas resep atau menyatakan secara tertulis.
6. Salinan resep harus di tandatangani oleh apoteker.
7. Resep harus dirahasiakan dan disimpan di Apotek dengan baik dalam jangka waktu 3 tahun. Resep atau salinan resep hanya dapat diperlihatkan kepada dokter penulis resep, atau yang merawat penderita, penderita yang bersangkutan, petugas kesehatan atau petugas lain yang berwenang menurut peraturan perundangan yang berlaku.

Dalam penyerahan obat kepada pasien harus diberi etiket yang dilekatkan pada wadah yang tertera sebagai berikut:

1. Nama Pasien
2. Aturan Pakai
3. Untuk obat yang melalui mulut perludi sebut sebagai obat dalam, memakai etiket kertas berwarna putih dan bagi obat luar yang tidak ditelan digunakan etiket kertas berwarna biru.

2.1.2 Resep

Resep merupakan permintaan tertulis dari seorang dokter kepada apoteker untuk membuat dan atau menyerahkan obat kepada pasien.

Dalam resep harus ditulis dengan menyebutkan

1. Tanggal dan tempat ditulisnya resep.
2. Nama Pasien
3. Aturan pakai obat

Terdapat tanda-tanda yang ditulis sebagai status dari resep tersebut, contoh tanda yang dituliskan pada bagian atas blanko resep sebagai berikut:

1. Statium/urgent = penting
2. Cito = segera
3. Ix3 = resep dapat diulang sebanyak 3x
4. N.I = resep tidak dapat di ulang

Jika terdapat kesalahan pada resep apoteker harus memberi tahu secara lisan atau tertulis kepada dokter yang menulis resep dan tidak boleh memberi tahukan pada pasien mengenai adanya kesalahan tersebut.

Selain obat yang sudah jadi atau paten, dokter berhak memodifikasi obat yang disebut obat racikan. Faktor-faktor yang berkaitan dengan obat ini adalah[3]:

1. Berat badan. Dosis rang yang kurang berat badannya adalah lebih kecil atau ditentukan dalam mg/Kg berat badan pasien.
2. Umur. Ada beberapa hal yang mempengaruhi adsorpsi, distribusi, metabolisme dan ekskresi obatpada bayi yang baru lahir:
    a. Beberapa sistem enzim pada bayi belum berkembang sempurna, sistem metabolisme obat dalam saluran pencernaan, fungsi hati dan ginjal baru berkembang setelah satu bulan, akibatnya:
        i. adsorpsi obat berjalan lambat.
        ii. timbul retensi obat di dalam badan.
    b. Fungsi ginjal belum sepenuhnya berkembang.
    c. Persentase air badan total dari berat badan total lebih besar dibandingkan anak yang lebih tua. Oleh karena itu volume distribusi obat pada bayi lebih besar dari pada anak yang lebih tua.
3. Jenis kelamin. Wanita lebih peka terhadap efek katartik tertentu daripada pria.
4. Kondisi patologik pasien.
    a. Penderita hipokalemia lebih peka terhadap digitalis dibandingpasien yang keadaan darah kaliumnya normal.

b. Penderita hipertiroid memerlukan dosis Luminal yang lebih tinggi untuk memperoleh efek peredaran daripada orang normal.
c. Penderita yang lebih peka terhadap suatu obat.

5. Idiosinkrasi. Merupakan respon abnormal yang sukar dijelaskan.

2.1.3 Hubungan antara Apotek dengan Asuransi Kesehatan

Asuransi kesehatan mungkin adalah asuransi yang paling penting untuk Anda miliki. Biaya pelayanan kesehatan yang mahal akan sangat membebani jika Anda tidak dijamin oleh asuransi. Tidak jarang ada orang yang harus menjual harta bendanya untuk menanggung biaya perawatan di rumah sakit. Anda tentu tidak harus seperti itu dengan memiliki asuransi.

Asuransi kesehatan di Indonesia saat ini disediakan baik oleh Pemerintah maupun swasta. Pemerintah melalui program Askeskin (asuransi kesehatan masyarakat miskin) menjamin biaya kesehatan sampai jumlah tertentu bagi para penduduk miskin pemegang kartu Askeskin. Para pegawai negeri juga dijamin biaya kesehatannya oleh Pemerintah melalui PT Askes yang preminya dibayar melalui pemotongan gaji PNS[4].

Bila Anda bukan penduduk miskin dan bukan pegawai negeri, Anda dapat membeli program asuransi komersial yang banyak diselenggarakan oleh perusahaan swasta nasional maupun asing. Bahkan, PT Askes juga menyediakan program “kepesertaan sukarela” bagi para pegawai di sektor swasta. Asuransi kesehatan komersial dapat dibeli oleh individu maupun kelompok (kumpulan). Karena pertimbangan administratif dan risiko, kebanyakan produk asuransi kesehatan hanya boleh dibeli oleh kelompok, bukan orang per orang.

Dibawah ini merupakan cara klaim asuransi terhadap apotek maupun rumah sakit, ada dua cara yang dapat dilakukan jika kita meminta klaim pada pihak asuransi[5]. Gambar 2.1 merupakan cara provider yang mengharuskan kita dirawat pada rumah sakit yang telah mempunyai ikatan kerjasama dengan pihak asuransi. Tetapi kita tidak mengeluarkan uang untuk biaya rumah sakit terlebih dahulu.

Gambar 2.1 Klaim asuransi dengan cara provider [5]

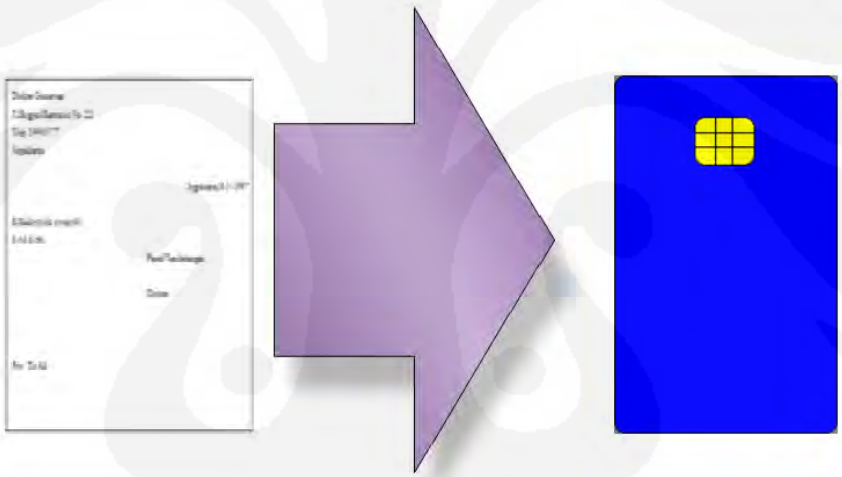
Gambar 2.2 merupakan cara yang kedua yaitu reimbursement dengan cara ini mungkin lebih lama kita akan mendapat uang penggantian dari pihak asuransi dan kita harus membayar segala biaya terlebih dahulu tetapi kita dapat memilih rumah sakit sesuai keinginan kita.

Gambar 2.2 Klaim Asuransi dengan cara Reimbursement [5]

Tidak semua asuransi mengaplikasikan kedua cara tersebut, sebagian hanya menggunakan salah satunya saja, baik cara provider maupun reimbursement. Pada tugas akhir saya ini tidak membahas asuransi lebih lanjut. Saya menganggap pasien tersebut menggunakan cara provider agar lebih mudah dalam pengaplikasiannya.

2.1.4 Hubungan *Smart Card* dengan Sistem Apotek

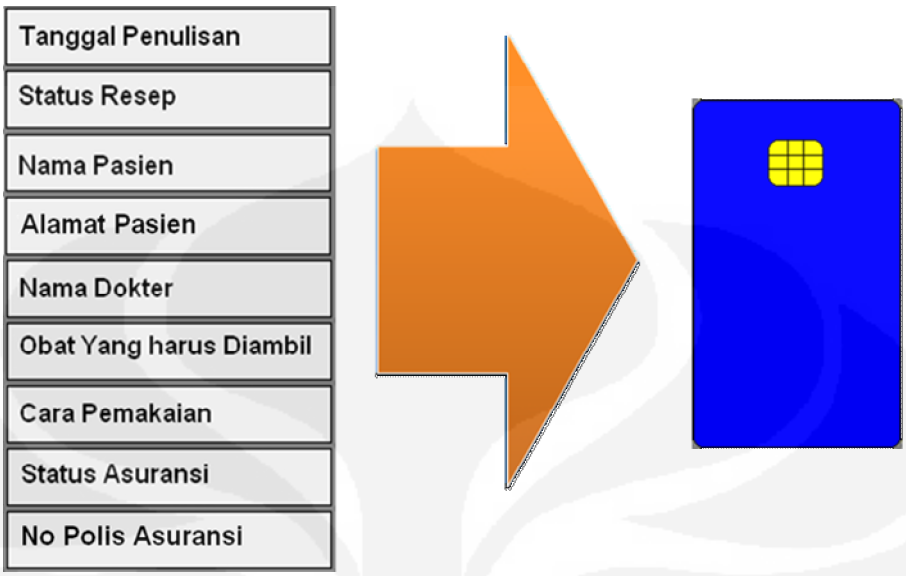
Pada smart apotek yang terjadi pada saat ini mengacu pada pencarian obat dari database secara manual tidak secara langsung. Itu dikarenakan resep yang di gunakan masih menggunakan cara konvensional yang masih menggunakan kertas. Oleh karena itu digantikanlah posisi resep tersebut dengan menggunakan *smart card* (Gambar 2.3).



Gambar 2.3 Penggantian resep dokter dengan kartu

Sehingga ketika kartu dimasukkan apoteker sudah dapat melihat obat apa saja yang harus diambil oleh pasien tersebut. Serta melihat apakah obat tersebut tersedia pada apotek tersebut apa tidak sehingga dapat mempercepat terjadinya transaksi pada apotek itu sendiri.

Pada Gambar 2.4 *applet* yang dimasukkan pada *smart card* tersebut terdapat asuransi yang tidak terdapat pada kertas resep, itu dikarenakan *smart card* ini juga berfungsi sebagai kartu pasien rumah sakit sehingga kita tidak perlu membawa kartu asuransi ke apotek ketika kita ingin menebus obat yang kita ambil.

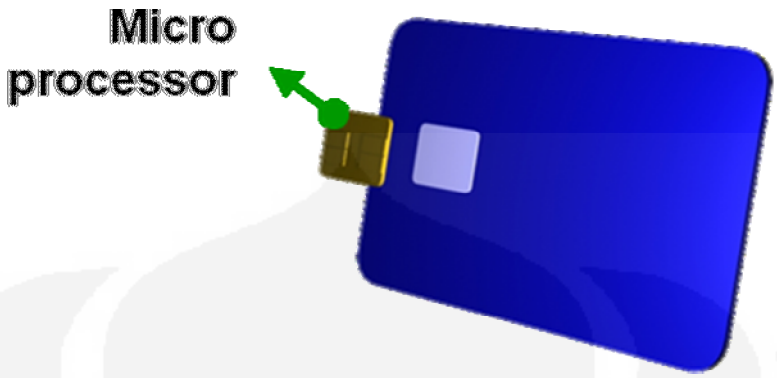


Gambar 2.4 *Applet* yang dimasukkan ke dalam kartu

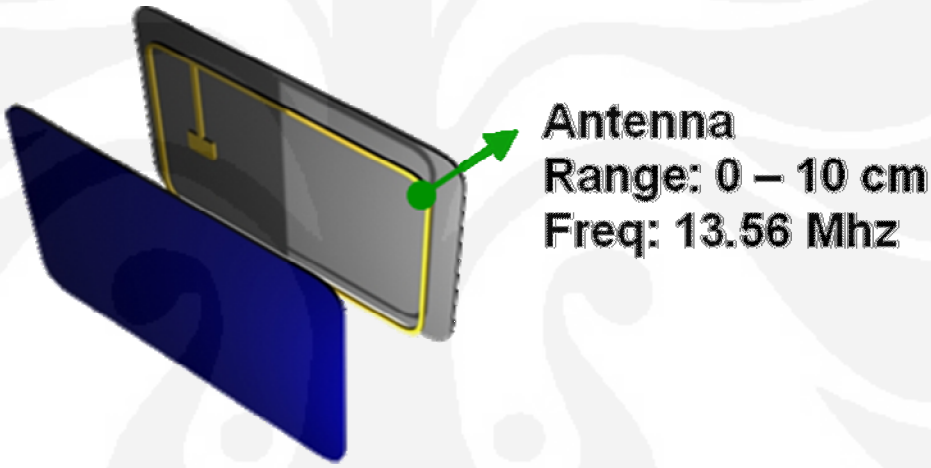
2.2 *SMART CARD*

Smart card adalah kartu plastik yang berisi sebuah *embedded* IC *(Integrated Circuit),* menyerupai kartu kredit atau ATM [6]. Bila dipakai untuk kartu SIM pada telepon selular, ukuran kartu plastik ini lebih kecil agar dapat dimasukkan ke dalam telepon selular. *Smart card* sangat aman secara desiannya dan terlindung dari interferensi agar informasi yang ada didalamnya tidak rusak atau hilang. *Smart card* tidak memiliki sumber daya sendiri, akan aktif hanya bila terhubung dengan *card reader*. Ketika dihubungkan, setelah melakukan proses *reset, smart card* akan tetap pasif menunggu *command request* dari aplikasi *client (host)*.

Smart card dapat berupa *contact* maupun *contact less*. *Smart card contact* bekerja dengan berkomunikasi melalui kontak fisik antara *card reader* dan 8 kontak pin yang terdapat pada *smart card*. Sementara *smart card contact less* berkomunikasi melalui sinyal frekuensi radio dengan jarak sekitar 60 cm [6]. Komunikasi radio pada *smart card contact less* berdasarkan teknologi yang menyerupai *tag* RFID *(Radio Frequency ID)* yang biasa ditemukan di toko-toko untuk kontrol persediaan dan mencegah pencurian. Gambar 2.5 dan Gambar 2.6 memperlihatkan bentuk fisik kedua jenis *smart card* tersebut.



Gambar 2.5 Bentuk fisik *smart card Contact.* [7]



Gambar 2.6 Bentuk fisik *smart card Contact less.* [7]

Java Card adalah sebuah kartu *Smart Card* yang dapat menjalankan program-program java. Tantangan terbesar dari disain teknologi *Java Card* adalah menyelaraskan *java system software* di dalam *Smart Card* dengan keterbatasan ruang untuk aplikasi. Solusinya adalah dengan hanya men-support sebuah subset dari fitur-fitur *java* dan untuk mengaplikasikan sebuah model terpisah untuk diimplementasikan ke *java virtual machine* (JVM).

Teknologi *Java Card* sendiri ada dalam bentuk selain *smart card* antara lain adalah *smart button* (kancing pintar) dan *USB token* (token USB) seperti terlihat pada Gambar 2.7 dan Gambar 2.8. Keduanya dapat digunakan seperti halnya *smart card*, contohnya untuk otentifikasi pengguna dan membawa

informasi. *Smart button* memiliki baterai dan berbasis kontak, sementara *USB token* dapat dimasukkan langsung ke *port* USB tanpa memerlukan *contact* atau *contact less reader*. Keduanya menyediakan kemampuan pemrograman dan memiliki properti yang sama dengan *smart card*.

Gambar 2.7 S*mart button* [6]

Gambar 2.8 *USB token* [6]

Dalam beberapa area, penggunaan *smart card* hanya sebagai kartu memori yang hanya memberikan media penyimpanan *non-volatile* yang terproteksi. *Smart card* yang lebih canggih memiliki mikroprosesor dan memori, untuk proses dan penyimpanan yang aman, dan dapat digunakan untuk aplikasi keamanan yang mengunakan *public-key* atau *shared-key algorithm*. Memori *non-volatile* pada *smart card* merupakan sumber yang sangat berharga karena dapat dipakai untuk menyimpan kunci rahasia dan sertifikat digital.

2.3 TEKNOLOGI JAVA CARD

Bertahun-tahun lalu, *Sun Microsystems* menyadari potensi dari *smart card* dan divais-divais yang memiliki keterbatasan sumber daya lainnya. Maka didefenisikanlah suatu *set* spesifikasi untuk *subset* dari teknologi *Java* yang digunakan untuk membuat aplikasi pada divais-divais tersebut, yang dikenal sebagai *Java Card applet*. Divais yang mendukung spesifikasi-spesifikasi ini

dirujuk sebagai *platform Java Card*. Pada *platform Java Card*, banyak aplikasi dari *vendor* yang berbeda dapat ada bersama-sama secara aman. Teknologi *Java Card* mengadaptasi *platform Java* untuk digunakan pada *smart card* dan divais lain yang lingkungannya sangat spesifik, dan kemampuan proses dan memori yang terbatas dibandingkan divais-divais pada *Java Mobile Environment* (J2ME).

Tipikal divais *Java Card* memiliki 8-bit atau 16-bit CPU berkecepatan 3,7Mhz, 1K RAM, dan lebih dari 16K *non-volatile* memori (EEPROM atau *flash*). Sementara *smart card* dengan unjuk kerja tinggi dilengkapi dengan sebuah prosesor terpisah dan *chip cryptographic* dan memori untuk enkripsi, beberapa diantaranya memiliki 32-bit CPU.

Spesifikasi teknologi *Java Card* yang sekarang merupakan versi 2.2, terdiri dari tiga bagian utama [8]:

1. Java Card Virtual Machine specification.

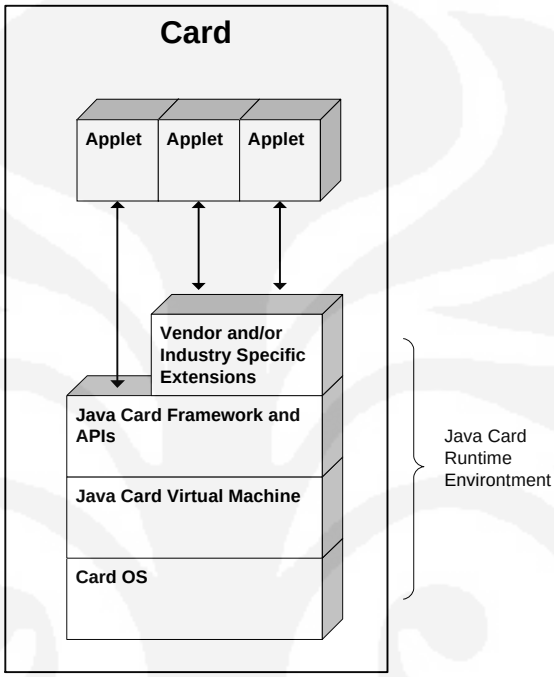
Mendefenisikan sebuah *subset* dari bahasa pemrograman *Java* dan VM untuk *smart card*. *Virtual Machine* untuk *platform Java Card* diimplementasikan dalam dua bagian, dengan satu bagian diluar kartu dan bagian yang lain dijalankan didalam kartu. *On-card* JCVM menginterpretasikan *field code*, mengatur *class* dan *object*, dan sebagainya. Bagian *Java* VM eksternal adalah *tool* pengembangan, biasanya disebut *tool Java Card Converter* yang mengambil, memverifikasi, dan persiapan lanjutan *class Java* pada *applet* kartu untuk eksekusi *on-card*. Keluaran dari *converter tool* ini adalah file *Converted Applet* (CAP) yang berisi semua *class* dalam sebuah paket (*package*) *Java* yang dapat di*load*, dan merupakan representasi *binary* yang dapat dieksekusi.

JCVM hanya mendukung *subset* yang terbatas dari bahasa pemrograman *Java* meskipun masih menyisakan fitur-fitur familiar seperti *object, inheritance, package, dynamic object creation, virtual method, interface dan exception.*

2. *Java Card Runtime Environment specification.*

Arsitektur *Java Card Runtime Environment* (JCRE) seperti terlihat pada Gambar 2.9. JCRE mendefenisikan *life-cycle* dari *Java Card* VM, *life-cycle applet*, bagaimana *applet* dipilih dan diisolasi satu sama lainnya, transaksi,

persistensi dan *sharing object*. JCRE menyediakan *interface* yang tidak tergantung *platform* kepada *service* yang disediakan oleh sistem operasi kartu. Terdiri dari *Java Card Virtual Machine*, *Java Card* API, dan *extend* khusus dari *vendor*.

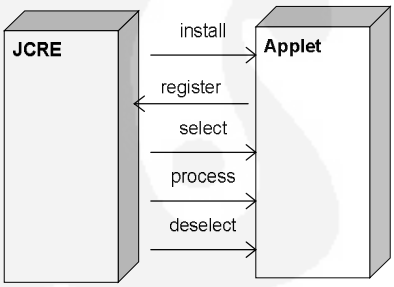


Gambar 2.9 Arsitektur *Java Card* dan JCRE [6]

- *Live-Cycle* dari *Java Card* VM. Usia JCVM dimulai hampir bersamaan dengan kartu itu sendiri, beberapa saat setelah kartu dibuat dan diuji, dan sebelum diberikan kepada pemegang kartu, dan akan berakhir bila kartunya dihancurkan. JCVM tidak akan berhenti meskipun tidak ada catu daya dan keadaannya disimpan di dalam memori *non-volatile* dari kartu. Ketika JCVM dimulai, akan menginisialisasi JCRE dan membuat semua *object* JCRE *framework* yang akan hidup selama umur JCVM. Setelah JCVM dimulai, semua interaksi dengan kartu, secara prinsipnya, diatur oleh satu dari *applet* yang ada dalam kartu. Ketika catu daya dilepas dari kartu, data yang berada di RAM akan hilang, sementara data yang disimpan di memori *non-volatile* akan tetap ada. Ketika kartu diberikan catu daya lagi, maka VM akan kembali

aktif, keadaan dari VM dan *object* akan dikembalikan, eksekusi sebelumnya dilanjutkan dan menunggu input selanjutnya.

- *Life-Cycle* dari *Java Applet*. Setiap *applet* dalam kartu secara unik diidentifikasikan oleh sebuah *Application ID* (AID). Suatu AID didefenisikan oleh ISO7816-5, adalah suatu sekuen antara 5 sampai 16 *field*. Semua *applet* harus merupakan *extend* dari *base class* abstrak *Applet*, yang mendefenisikan *method-method* yang digunakan oleh JCRE untuk mengontrol *life-cycle applet* seperti diperlihatkan Gambar 2.9. *Life-Cycle Java Applet* bermula ketika *applet* di*download* (atau di*upload*, tergantung melihatnya dari sisi mana) kedalam kartu dan JCRE memanggil *method static applet*: `applet.install()`, dan *applet* akan *register* sendiri dengan JCRE dengan memanggil `applet.register()`. Ketika *applet* di*install* dan di*register*, keadaannya tidak terseleksi *(unselected)*, siap untuk menerima pemilihan dan proses APDU. Gambar 2.10 memperlihatkan operasi dari *method applet*



Gambar 2.10 Operasi dan *method applet* [6]

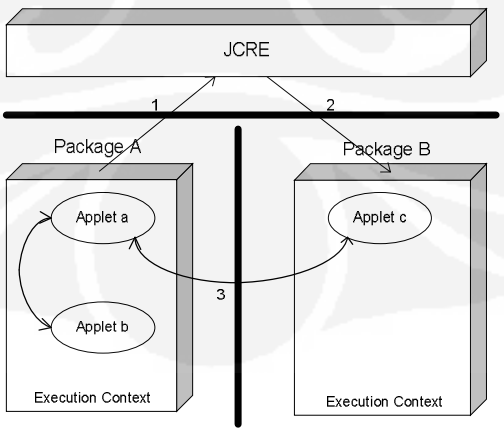
Ketika dalam keadaan *unselected*, *applet* tidak aktif. *Applet* dipilih *(selected)* untuk pemrosesan APDU apabila aplikasi *host* meminta JCRE untuk memilih suatu *applet* tertentu dalam kartu dengan instruksi `SELECT APDU` atau `MANAGE CHANNEL APDU`. Untuk memberitahu *applet* yang telah dipilih oleh applikasi *host*, JCRE memanggil *method* `select()`, *applet* secara tipikal akan melakukan inisialisasi yang sesuai dalam persiapan untuk pemrosesan APDU.

- Sesi *Java Card* dan Kanal Logika.

Sesi kartu *(Card Session)* adalah perioda sejak kartu diberi catu daya dan saling bertukar APDU dengan *card reader*. *Java Card* mendukung konsep kanal logika *(logical channel)* yang mengizinkan sampai 16 sesi aplikasi kedalam *smart card* untuk dibuka dalam waktu yang sama, dengan satu sesi setiap kanal logikanya.

- Isolasi *Applet* dan Pembagian *(Sharing) Object.*

  Platform Java Card merupakan lingkungan multi-aplikasi yang aman dimana banyak *applet* dari *vendor* yang berbeda dapat secara aman berada bersama-sama dalam kartu. Tiap *applet* diberikan kepada sebuah *execution context* yang mengontrol akses ke *object* yang diberikan padanya. Batas antara satu *execution context* dengan yang lainnya sering disebut sebagai *applet firewall* yang merupakan pengembangan *Java Card Runtime* dari konsep keamanan *Java sandbox,* dan mengkombinasikan dengan fungsi *class loader* (`java.ClassLoader`), dan pengontrol akses (`java.AccesController`). *Firewall Java Card* menciptakan sebuah *heap virtual* (lokasi pemori pada pemrograman *Java*) seperti halnya sebuah *object* dapat mengakses *method (public)* dan data hanya bila *object-object* tersebut berada dalam satu *firewall* yang sama, seperti diilustrasikan pada Gambar 2.11. *Platform Java Card* mendukung sharing *object* yang aman antar *firewall*.



Gambar 2.11 *Firewall* pada *Java Card* [6]

3. *Java Card* API *specification*, yang mendefenisikan suatu *subset* kecil dari *Application Programming Interface* (API) dari bahasa pemrograman *Java* dan menambahkan *Java Card Framework* yang mendefenisikan *set*nya sendiri dari *class* inti (*core classes*) yang secara khusus mendukung aplikasi *Java Card*. *Java Card* API ini memiliki *package-package* sebagai berikut:

   - `java.io` yang mendefenisikan suatu `exception class`, merupakan *base class* `IOException`, untuk melengkapi `exception hierarchy rmi`. Tidak ada `java.io` dari bahasa pemrograman *Java* tradisional yang dimasukkan.
   - `java.lang` yang mendefenisikan *class* `Object` dan `Throwable` dan mendefenisikan sejumlah *class* `exception`; *base class* `exception`, berbagai `runtime exception`, dan `CardException`. Tidak ada `java.lang` dari bahasa pemrograman *Java* tradisional yang dimasukkan.
   - `java.rmi` mendefenisikan *interface* `Remote` dan class `RemoteException`. Tidak ada `java.rmi` dari bahasa pemrograman *Java* tradisional yang dimasukkan.
   - `javacard.framework` yang mendefenisikan *interface*, *class*, dan *exception* yang membentuk *core* dari *Java Card Framework* seperti pada Tabel 2.1. *Package* `javacard.framework` mendefenisikan konsep-konsep penting seperti *Personal Identification Number* (PIN), *Application Protocol Data Unit* (APDU), *applet Java Card (Applet)*, *Java Card System (JCSystem)* dan *class utility*. `javacard.framework` juga mendefinikan berbagai konstanta ISO7816 dan berbagai *exception* yang spesifik dari *Java Card* [9].

Tabel 2.1 Isi dari *package* javacard.framework[6]

Interface	ISO7816 defines constants related to ISO 7816-3 and ISO 7816-4. MultiSelectable identifies applets that can support concurrent selections. PIN represents a personal identification number used for security (authentication) purposes. Shareable identifies a shared object. Objects that must be available through the applet firewall must implement this interface.
Classes	AID defines an ISO7816-5-conforming Application Identifier associated with an application provider; a mandatory attribute of an applet. APDU defines an ISO7816-4-conforming Application Protocol Data Unit, which is the communication format used between the applet (on-card) and the host application (off-card). Applet defines a Java Card application. All applets must extend this abstract class. JCSystem provides methods to control the applet life-cycle, resource and transaction management, and inter-applet object sharing and object deletion. OwnerPIN is an implementation of the PIN interface. Util provides utility methods for manipulation of arrays and shorts, including arrayCompare(), arrayCopy(), arrayCopyNonAtomic(), arrayFillNonAtomic(), getShort(), makeShort(), setShort().
Exceptions	Various Java Card VM exception classes are defined: APDUException, CardException, CardRuntimeException, ISOException, PINException, SystemException, TransactionException, UserException.

- *java*card.framework.*service* yang mendefenisikan *interface*, *class*, dan *exception* untuk *service*. *Service* memproses *incoming command* (perintah yang datang) dalam format APDU seperti pada Tabel 2.2.

Tabel 2.2 Isi dari *package* javacard.framework.service [6]

Interface	Service, the base service interface, defines the methods processCommand(), processDataIn(), and processDataOut(). RemoteService is a generic Service that gives remote processes access to services on the card. SecurityService extends the Service base interface, and provides methods to query the current security status, including isAuthenticated(), isChannelSecure(), and isCommandSecure().
Classes	BasicService is a default implementation of a Service; it provides helper methods to handle APDUs and service collaboration. Dispatcher maintains a registry of services. Use a dispatcher if you want to delegate the processing of an APDU to several services. A dispatcher can process an APDU completely with the process() method, or dispatch it for processing by several services with the dispatch() method.
Exceptions	ServiceException a service-related exception.

- *java*card.security yang mendefenisikan *interface* dan *class* untuk *Java Card security framework* seperti terlihat pada Tabel 2.3.

Tabel 2.3 Isi dari *package* `javacard.security[6]`

`Interface`	`Generic base interfaces Key, PrivateKey, PublicKey, and SecretKey, and subinterfaces that represent various types of security keys and algorithms: AESKey, DESKey, DSAKey, DSAPrivateKey, DSAPublicKey, ECKey, ECPrivateKey, ECPublicKey, RSAPrivateCrtKey, RSAPrivateKey, RSAPublicKey`

`Classes`	`Checksum: abstract base class for CRC algorithms`
	`KeyAgreement: base class for key-agreement algorithms`
	`KeyBuilder: key-object factory`
	`KeyPair: a container to hold a pair of keys, one private, one public`
	`MessageDigest: base class for hashing algorithms`
	`RandomData: base class for random-number generatorss`
	`Signature: base abstract class for signature algorithms`

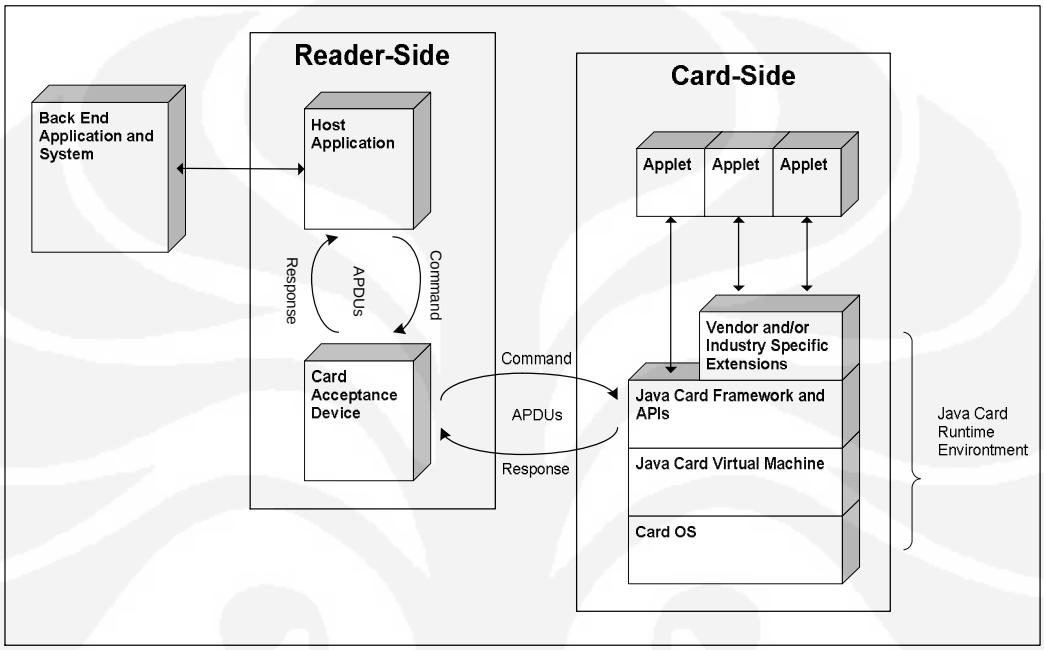
`Exceptions`	`CryptoException: encryption-related exceptions such as unsupported algorithm or uninitialized key.`

- `javacardx.crypto` merupakan *extend package* yang mendefenisikan *interface* `KeyEncryption` dan *class* `Chiper`.
- `javacardx.rmi` merupakan suatu *extend package* yang mendefenisikan *class* RMI yaitu *class* `CardRemoteObject` dan `RMIService`. `CardRemoteObject` memiliki *method* `export()` dan `unexport()` untuk *enable* dan *disable remote acces* terhadap suatu *object* dari luar kartu. `RMIService` merupakan *extend* dari `BasicService` dan *implement* `RemoteService` untuk memproses permintaan RMI.

Sun Microsystems juga menyediakan *Java Card Development Kit* (JCDK), yang meliputi suatu referensi dari implementasi *Java Card RE* dan *Java Card* VM, dan *tool* lain yang membantu pengembangan *Applet Java Card.*

2.3.1 Elemen-elemen dari Aplikasi Java Card

Aplikasi *Java Card* bukan merupakan aplikasi yang berdiri sendiri, tetapi meliputi aplikasi *card side, reader side,* dan *back end* seperti diperlihatkan pada Gambar 2.12.



Gambar 2.12 Elemen-elemen aplikasi Java Card [6]

Sebuah aplikasi lengkap *Java Card* terdiri dari aplikasi *back end* dan sistem, aplikasi *host (off card), interface device (card reader)*, dan *applet on-card*, informasi dari user dan perangkat lunak penunjang.

1. Aplikasi *Back end* dan Sistem.

   Aplikasi *back-end* menyediakan layanan yang mendukung *in-card Java applet*. Contohnya; aplikasi *back end* dapat menyediakan keterhubungan dengan dengan sistem keamanan dimana secara bersama-sama dengan informasi dari user akan menghasilkan keamanan yang kuat.

2. Aplikasi *Host (Reader Side, off card).*

   Aplikasi *host* berada pada terminal seperti PC, terminal pembayaran elektronik, telepon selular, atau subsistem keamanan. Aplikasi *host* menangani komunikasi antar pengguna, *applet Java Card*, dan penyedia aplikasi *back end*.

Vendor smart card biasanya menyediakan kit pengembangan dan API *(Application Programming Interface)* untuk mendukung aplikasi *host* dan juga *applet Java Card*. Contoh-contoh menyertakan *OpenCard Framework*, suatu *set* dari API berbasis *Java* yang menyembunyikan beberapa detail dalam berinteraksi dengan *card reader* dari *vendor-vendor* yang berbeda, model *distributed-object* (object terdistribusi) dari *Java Card Remote Method Invocation* (JCRMI), dan *Security and Trust Service* API (SATSA).

3. *Card Acceptance Device (Reader-Side).*

   Card Acceptance Device (CAD) adalah divais antarmuka *(interface device)* antara aplikasi *host* dengan divais *Java Card*. CAD menyediakan daya ke kartu baik secara elektrikal maupun dengan komunikasi RF. Sebuah CAD dapat berupa sebuah *card reader* yang dihubungkan ke komputer personal dengan mengunakan *port serial*, atau bisa saja terintegrasi dengan sebuah terminal seperti terminal pembayaran elektronik di toko-toko, restoran dan sebagainya. *Interface device* mengirimkan perintah *Application Protocol Data Unit* (APDU) dari aplikasi *host* ke kartu, dan mengirimkan respon dari kartu ke aplikasi *host*. APDU akan dibahas lebih lanjut. Beberapa *CAD* memiliki *keyboard* untuk memasukkan PIN dan juga memiliki tampilan.

4. *On Card Applet* dan *Environment.*

   Platform Java Card merupakan lingkungan (*environment*) yang multi-aplikasi dimana satu atau lebih *applet Java Card* bisa disimpan dalam kartu, bersama dengan perangkat lunak pendukung seperti sistem operasi dari kartu dan *Java Card Runtime Environment* (JCRE). JCRE terdiri dari *Java Card* VM, *Java Card Framework* dan API, dan beberapa *extend* dari API.

   Semua *applet Java Card* yang berasal dari *base class* harus mengimplementasikan *(implement) method* `install()` dan `process()`. JCRE memunculkan `install()` ketika menginstal *applet*, dan `process()` setiap kali ada APDU yang datang ke *applet*.

   Applet Java Card akan aktif bila dipanggil dan akan tetap hidup meskipun sumber dayanya dimatikan. *Applet* kartu berperilaku seperti *server* dan bersifat pasif. Setelah kartu dinyalakan *(power up)*, tiap applet akan tetap tidak aktif sampai *applet* tersebut dipilih, inisialisasi dilakukan. *Applet* akan aktif

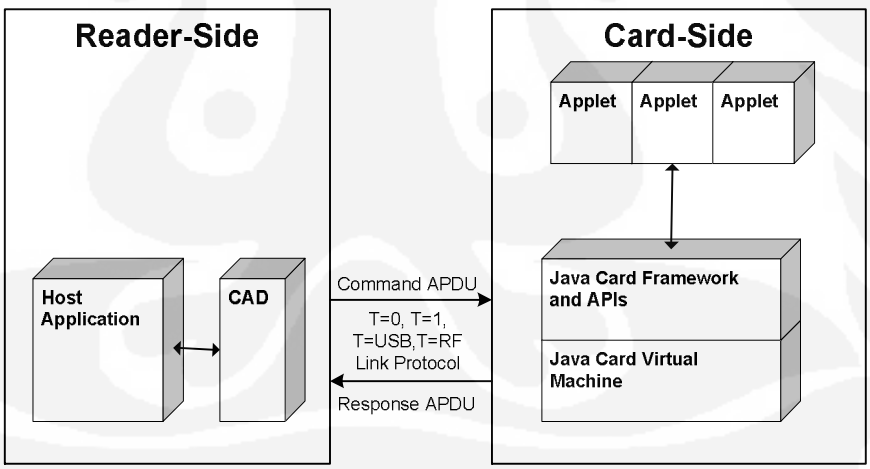
hanya bila sebuah APDU telah terkirim kepadanya. Bagaimana *applet* menjadi aktif telah dijelaskan sebelumnya pada bagian *Java Card Live-Cycle*[10].

2.3.2 Berkomunikasi dengan Applet Java Card

Ada dua model untuk berkomunikasi antara aplikasi *host* dengan *applet Java Card* yaitu model *message-passing* dan berbasis *Java Card Remote Method Invocation* (JCRMI) yang merupakan *subset* dari J2SE RMI model *object* terdistribusi *(distributed-object model)*[11].

1. Model *Message-Passing*

   Model *message-passing* yang diilustrasikan pada Gambar 2.13 merupakan dasar dari seluruh komunikasi *Java Card*. Ditengahnya adalah APDU, paket data logika yang saling bertukar antar *CAD* dan *Java Card Framework*. *Java Card* Framework menerima dan meneruskan ke *applet* yang sesuai semua *command* APDU yang dikirimkan oleh CAD. *Applet* akan memproses *command* APDU tersebut dan mengembalikan suatu *response* APDU. APDU mengikuti standar internasional ISO/IEC 7816-3 dan 7816-4.

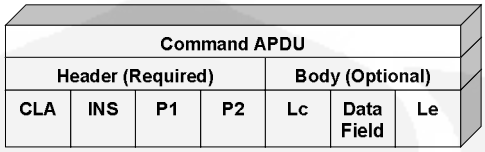


Gambar 2.13 Model *message-passing[6]*

Komunikasi antara *reader* dan kartu biasanya berdasarkan dua *link* protokol yaitu *field-oriented* (T=0) atau *block-oriented* (T=1). Protokol alternatif adalah T=USB dan T=RF dapat digunakan. *Class* JCRE APDU menyembunyikan beberapa detail protokol dari aplikasi, tapi tidak semuanya, karena protokol T=0 agak kompleks.

b. *Command* (Perintah) APDU

Struktur dari *command* APDU diatur oleh nilai *field* pertama dan dalam banyak kasus terlihat seperti Gambar 2.14



Gambar 2.14 Struktur dan *command* APDU [6]

Command APDU membutuhkan *header* dan *body* yang terdiri dari:

- CLA (1 *field*): *Field* yang diperlukan untuk mengidentifikasi dari suatu *class* aplikasi-spesifik dari instruksi. Nilai CLA yang valid didefenisikan dalam ISO 7816-4 seperti pada Tabel 2.4.

Tabel 2.4 Nilai CLA pada ISO 7816 [6]

Nilai CLA	***Class* Instruksi**
0x0n, 0x1n	ISO 7816-4 instruksi kartu, seperti akses file
20 sampai 0x7F	Dicadangkan
0x8n atau 0x9n	ISO/IEC 7816-4 format yang dapat digunakan untuk *application-specific instruction*
0xAn	*Application-specific instruction* atau *Vendor-specific instruction*
B0 sampai CF	ISO/IEC 7816-4 format yang dapat digunakan untuk *application-specific instruction*
D0 sampai FE	*Application-specific instruction* atau *Vendor-specific instruction*
FF	Dicadangkan untuk pemilihan tipe protokol

Secara teori nilai CLA 0x80 keatas bisa digunakan untuk instruksi-instruksi *application-specific*, tetapi dalam banyak implementasi *Java Card* dewasa ini, hanya yang dicetak tebal yang digunakan.

- INS (1 *field*): *Field* ini diperlukan untuk mengindikasikan suatu instruksi spesifik didalam *class* instruksi yang diidentifikasikan oleh *field* CLA. Standar ISO 7816-4 menspesifikasikan instruksi dasar untuk digunakan dalam mengakses data pada kartu bila distrukturkan sesuai dengan file sistem pada kartu *(on-card file system)* seperti yang didefenisikan pada standar. Fungsi-fungsi tambahan dispesifikasikan di bagian lain dari standar. Tabel 2.5. merupakan daftar dari beberapa instruksi ISO 7816. Kita dapat mendefenisikan INS *application-specific* sendiri hanya apabila menggunakan nilai CLA *field* yang sesuai dengan standar.

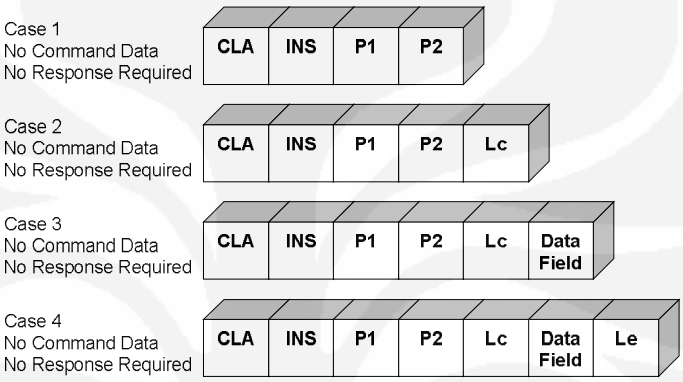
Tabel 2.5 Nilai INS pada ISO 7816 ketika CLA = 0X [6]

Nilai INS	Penjelasan Perintah
0E	Erase Binary
20	Verify
70	Manage Channel
82	External Authenticate
84	Get Challenge
88	Internal Authenticate
A4	Select File
B0	Read Binary
B2	Read Record(s)
C0	Get *Response*
C2	Envelope
CA	Get Data
D0	Write Binary
D2	Write Record
D6	Update Binary
DA	Put Data
DC	Update Record
E2	Append

- P1 (1 *field*): *Field* yang dibutuhkan untuk mendefenisikan parameter 1.
- P2 (1 *field*): *Field* yang dibutuhkan untuk mendefenisikan parameter 2.
- Lc (1 *field*): *Field* opsional adalah jumlah *field* pada *field* data dari *command*.

- Data *field* (variabel, banyak *field* Lc): *Field* opsional untuk menyimpan data dari *command*.
- Le (1 *field*): *Field* opsional yang menspesifikasikan banyak *field* maksimum pada *field* data dari respon yang diharapkan.

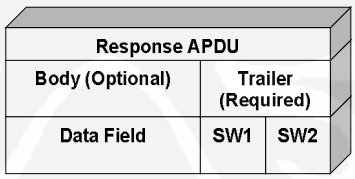
Tergantung dari keberadaan data *command* dan *response* yang diperlukan, ada 4 variasi dari *command* APDU bila menggunakan protokol T=0(Gambar 2.15).



Gambar 2.15 Variasi dari *command* APDU untuk protokol T=0 [6]

c. *Response* APDU

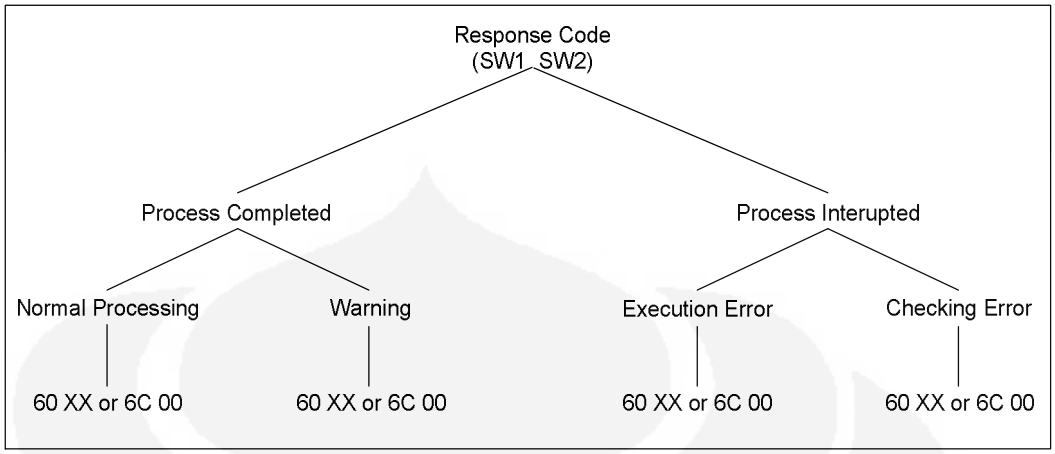
Format dari *Response* APDU (Gambar 2.16) lebih sederhana dan seperti *command* APDU, juga memiliki *field* yang harus ada dan *field* opsional yaitu.



Gambar 2.16 Format dari *response* APDU [6]

- Data *field* (panjang varibel, ditentukan oleh Le pada *command* APDU)
- SW1 (1 *field*): *Field* opsional yang merupakan *status word* 1.
- SW2 (1 *field*): *Field* opsional yang merupakan *status word* 2.

Gambar 2.17 menunjukkan nilai Status *Word* ditentukan oleh spesifikasi ISO 7816-4.



Gambar 2.17 Kode *response* status [6]

d. *Processing* APDU.

Setiap kali ada *incoming* APDU untuk suatu *applet* yang dipilih, JCRE memanggil *method applet* `process()`, melewatkan incoming APDU tadi sebagai *argument*. *Applet* harus menterjemahkan *command* APDU tadi, memproses data, membangkitkan *response* APDU dan mengembalikan kontrol kepada JCRE.

2. *Java Card Remote Method Invocation* (JCRMI).

Pada model RMI, sebuah aplikasi *server* menciptakan dan membuat *object* `remote` dan aplikasi *client* memberikan referensi `remote` kepada *object* `remote`, dan memanggil *method* `remote` untuk *object* `remote` tersebut. Pada JCRMI, *applet Java Card* merupakan *server* dan aplikasi *host* adalah *client*.

JCRMI disediakan dalam paket *extend* `javacardx.rmi` dengan *class* `RMIService`. *Message* JCRMI dienkapsulasi dalam *object* APDU yang dilewatkan ke *method* `RMIService`. Dengan kata lain JCRMI menyediakan sebuah mekanisme model *object* terdistribusi yang terletak diatas model *messaging* berbasis APDU, dimana *server* dan *client* berkomunikasi, melewatkan informasi *method*, *argument* dan *return value*.

BAB III
METODE PERANCANGAN IMPLEMENTASI JAVACARD PADA SISTEM DATABASE APOTEK

Metode perancangan menggunakan *use case* diagram, mendeskripsikan aktor-aktor dan hubungannya, sekuensial diagram menerangkan sekuensial dari program yang dibuat. Aplikasi *smart card* pada sistem database apotek mengunakan Java Card RMI dengan pendekatan model yang berpusat pada objek. Aplikasi dibuat menggunakan model RMI dengan membuat aplikasi *server*, remote objek dan aplikasi pengguna, dan membuat remote reference pada *server* remote objek.

Menggunakan pemograman berorientasi objek memberikan beberapa keuntungan, antara lain tidak perlu mengetahui tentang detail dari *smart card* dan *card reader*, tidak perlu mengetahui tentang *low leve*l komunikasi APDU, kode program lebih mudah untuk didisain dan dimaintain, dengan waktu pengembangan yang lebih singkat.

3.1 PRINSIP KERJA SISTEM

Prinsip kerja sistem apotek dimulai dari pembacaan database obat yang dapat diakses secara langsung ataupun dengan menggunakan *smart card* yang berisikan resep obat yang telah ditulis sebelumnya dari dokter. Pada sistem ini terdapat dua jenis pengguna yang memiliki otoritas akan program tersebut. Pengguna yang pertama saya sebut *administrator* dan yang kedua adalah *user*.

Administrator mampu mengakses kegiatan masuknya obat dan melakukan pengeditan database. Sedangkan *user* hanya mampu melayani jika ada pembeli yang ingin membeli obat.

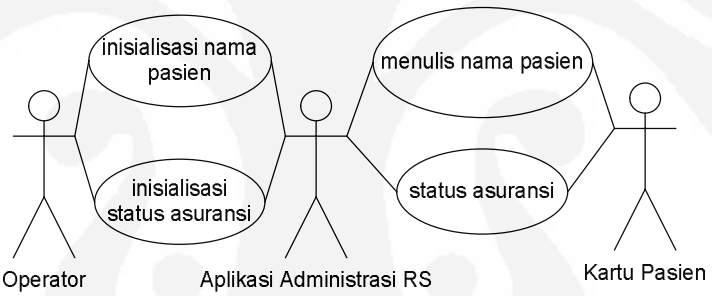
Resep disini menggunakan *smart card*, oleh karena itu harus ada program dokter yang berguna untuk menulis resep pada *smart card* tersebut dan terdapat program administrator rumah sakit untuk beberapa hal yang tidak dicantumkan oleh dokter tetapi harus ada pada kertas resep tersebut seperti nama pasien dan asuransi yang termasuk hal yang diproses pada program sistem apotek tersebut.

3.2 *USE CASE DIAGRAM* APLIKASI JAVACARD PADA APOTEK

Dalam aplikasi *Java Card* pada sistem apotek ini terdapat tiga aplikasi yang digunakan, yaitu administrasi rumah sakit, dokter dan apotek. Aplikasi tersebut merupakan simulasi sederhana dari aplikasi yang ada sebenarnya. *Use case* diagram digunakan untuk melihat hubungan para aktor pada suatu aplikasi.

3.2.1 *Use Case Diagram* Aplikasi Administrasi RS

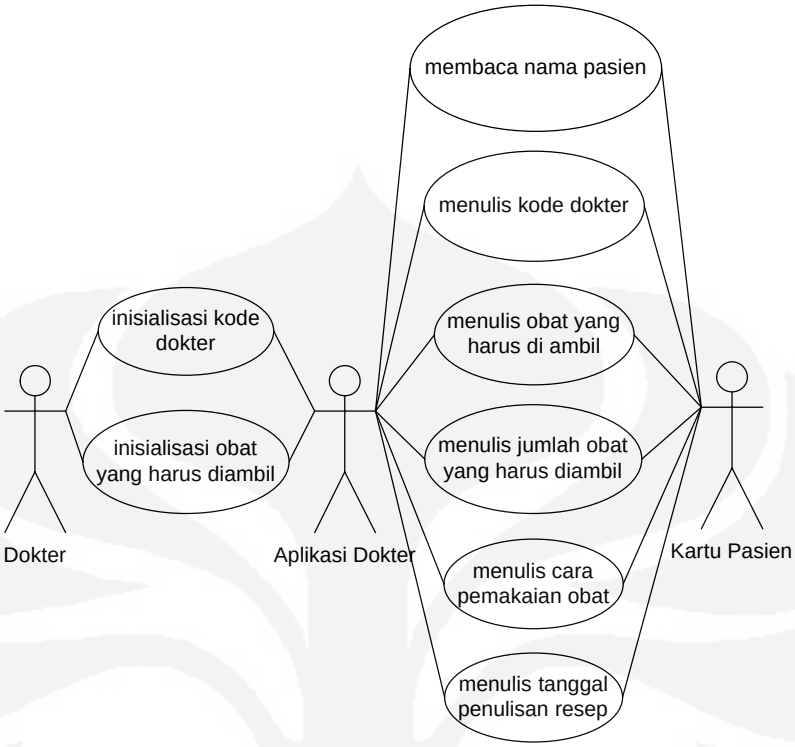
Aplikasi administrasi rumah sakit merupakan program sederhana yang digunakan oleh operator pada administrasi rumah sakit untuk mengisikan nama pasien dan status asuransi dari pasien tersebut pada kartu pasien. Dimana kartu pasien merupakan aplikasi *smart card* yang dapat di gunakan oleh si pasien itu sendiri. Kartu ini menyimpan data kode dokter yang merawat, obat yang harus diambil oleh pasien, nama pasien, dan status asuransi dari pasien tersebut.



Gambar 3.1 *Use case diagram* aplikasi administrasi RS

3.2.2 *Use Case Diagram* Aplikasi Dokter

Aplikasi dokter adalah program yang digunakan oleh dokter untuk dapat menulis resep pada *smart card*. Sebagaimana syarat resep yang mencantumkan tanggal pembuatan, nama dokter dan obat yang harus diambil beserta cara pemakaian obat akan dapat dituliskan pada program aplikasi dokter.



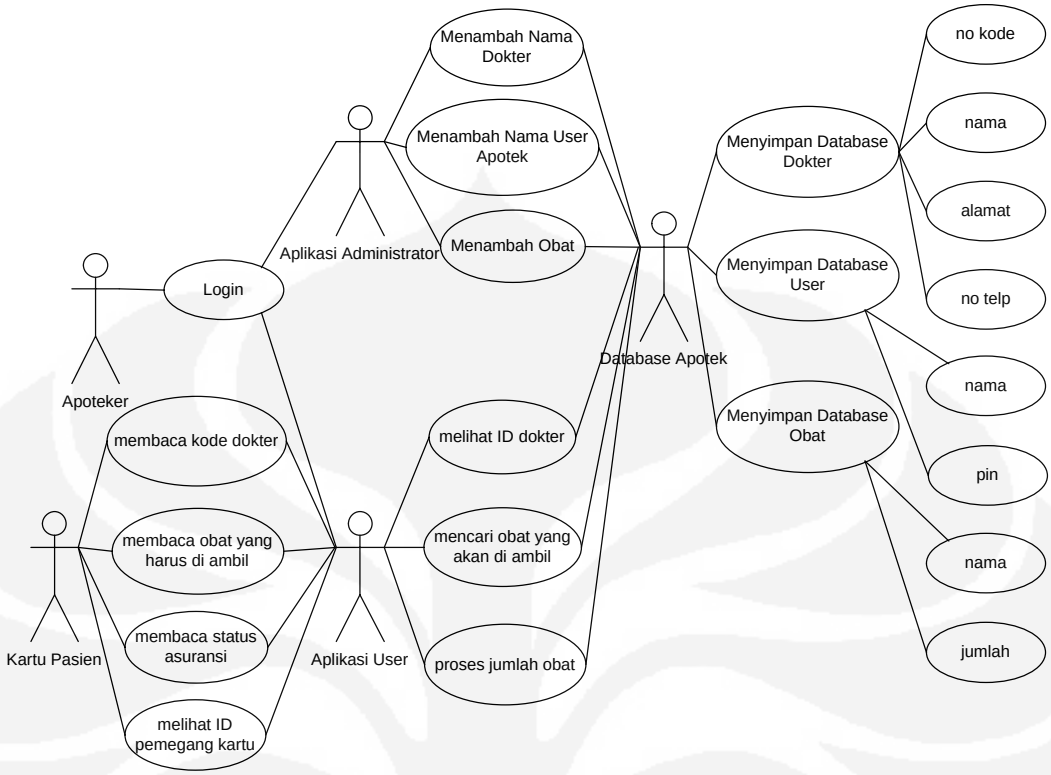
Gambar 3.2 *Use case diagram* aplikasi dokter

3.2.3 *Use Case Diagram* Aplikasi Apotek

Aplikasi apotek adalah program yang digunakan oleh apoteker untuk melakukan kegiatan dalam apotek. Aplikasi ini di bagi menjadi dua bagian, yakni aplikasi administrator dan user.

Administrator digunakan untuk menambah atau mengedit database pada apotek tersebut, seperti daftar karyawan atau user, menambah obat baik macam obat maupun jumlahnya, dan menambah daftar dokter.

Sedangkan user digunakan untuk komunikasi dengan pelanggan. Baik dengan menggunakan kartu maupun tidak menggunakan kartu atau langsung. Kita dapat melihat identitas dokter yang memberi obat, obat apa saja yang harus diambil oleh si pasien dan status asuransi pada kartu pasien. Aplikasi ini akan memproses obat jika stok obat yang ada dalam database masih ada. Status asuransi digunakan untuk perbedaan sistem pembayaran saja, untuk proses yang lainnya sama dengan yang tidak mempergunakan asuransi.



Gambar 3.3 *Use case diagram* aplikasi apotek

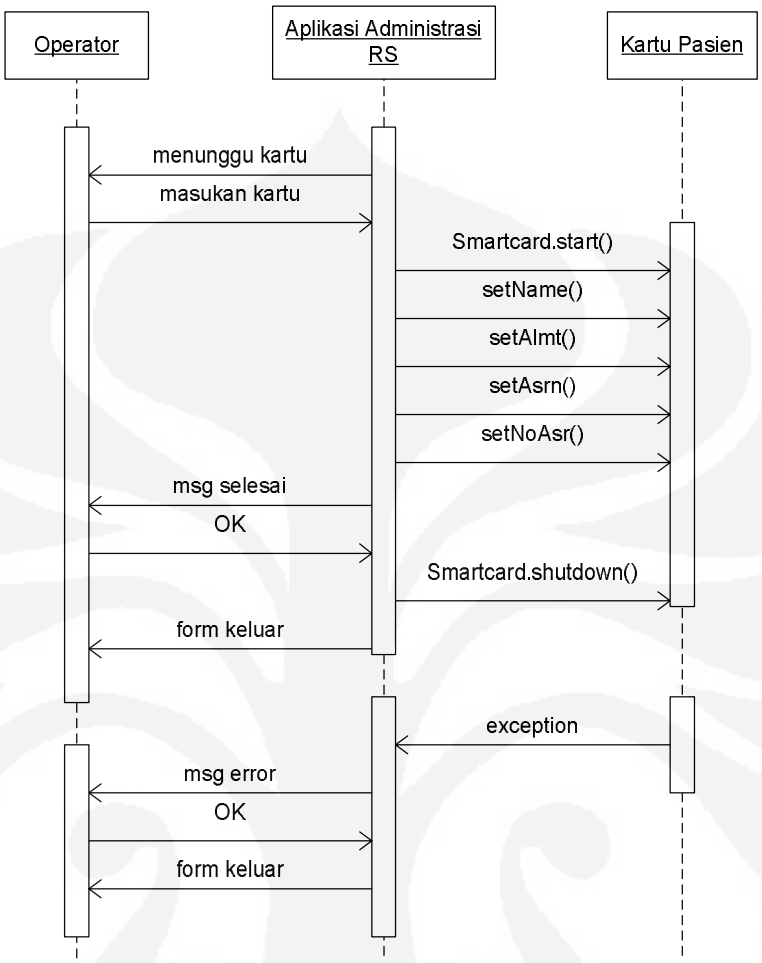
3.3 DIAGRAM SEKUENSIAL APLIKASI JAVACARD

Pada diagram sequensial kita dapat melihat kegiatan yang terjadi pada setiap program aplikasi dengan lebih jelas secara berurutan. Aplikasi ini dibagi menjadi beberapa sub sistem agar dapat terlihat lebih jelas jalannya aplikasi setiap program.

Pertama-tama, aplikasi administrator akan menunggu kartu untuk di masukkan oleh operator. Setelah kartu di masukkan maka program aplikasi akan melihat inisialisasi awal pada kartu yang akan di tulis itu benar apa tidak.

Jika tidak maka program akan menampilkan form keluar. Jika benar maka kartu akan dapat di tulis dengan nama pasien tersebut dan status asuransi dari pasien tersebut untuk digunakan pada aplikasi selanjutnya.

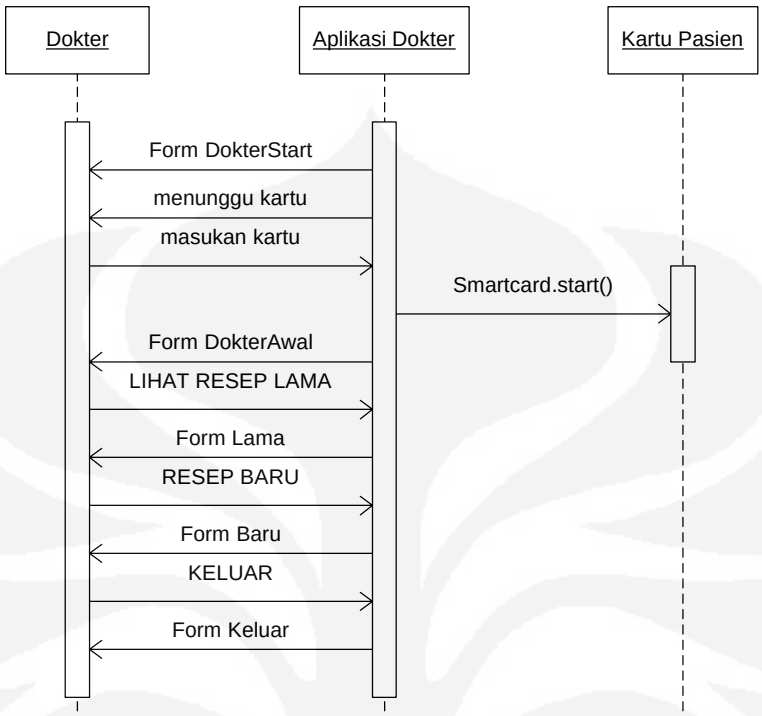
Jika sudah selesai akan keluar pesan selesai, lalu jika kita menekan OK kartu *smart card* akan di tutup dan menampilkan form keluar. Kita dapat melihat diagram sekuensialnya pada gambar 3.4 dibawah ini.



Gambar 3.4 Diagram sekuensial aplikasi administrasi RS

Pada aplikasi dokter, awalnya akan menunggu kartu untuk di masukkan oleh dokter. Setelah kartu di masukkan, maka program aplikasi akan melihat inisialisasi awal pada kartu yang akan di tulis itu benar apa tidak.

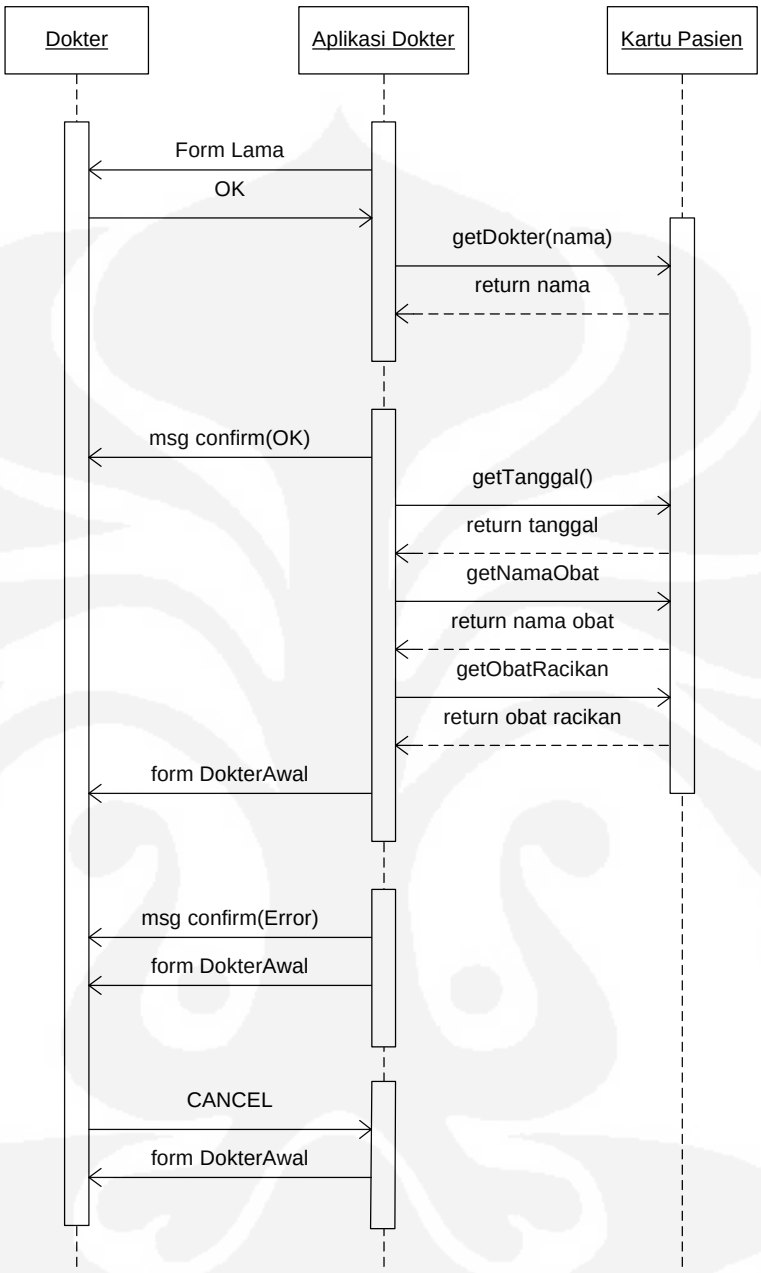
Jika tidak, maka program akan menampilkan form keluar. Jika benar maka dokter dapat memilih apa yang akan dilakukan, apakah ingin melihat resep sebelumnya atau langsung menulis resep baru. Jika penulisan sudah selesai akan keluar pesan selesai, lalu jika kita menekan OK kartu *smart card* akan di tutup dan menampilkan form keluar.



Gambar 3.5 Diagram sekuensial aplikasi dokter awal

Jika dokter memilih melihat resep lama, maka nama dokter sebelumnya akan dicocokkan dengan dokter yang sekarang. Karena resep obat hanya dapat dilihat oleh dokter yang sama yang menulis resep tersebut.

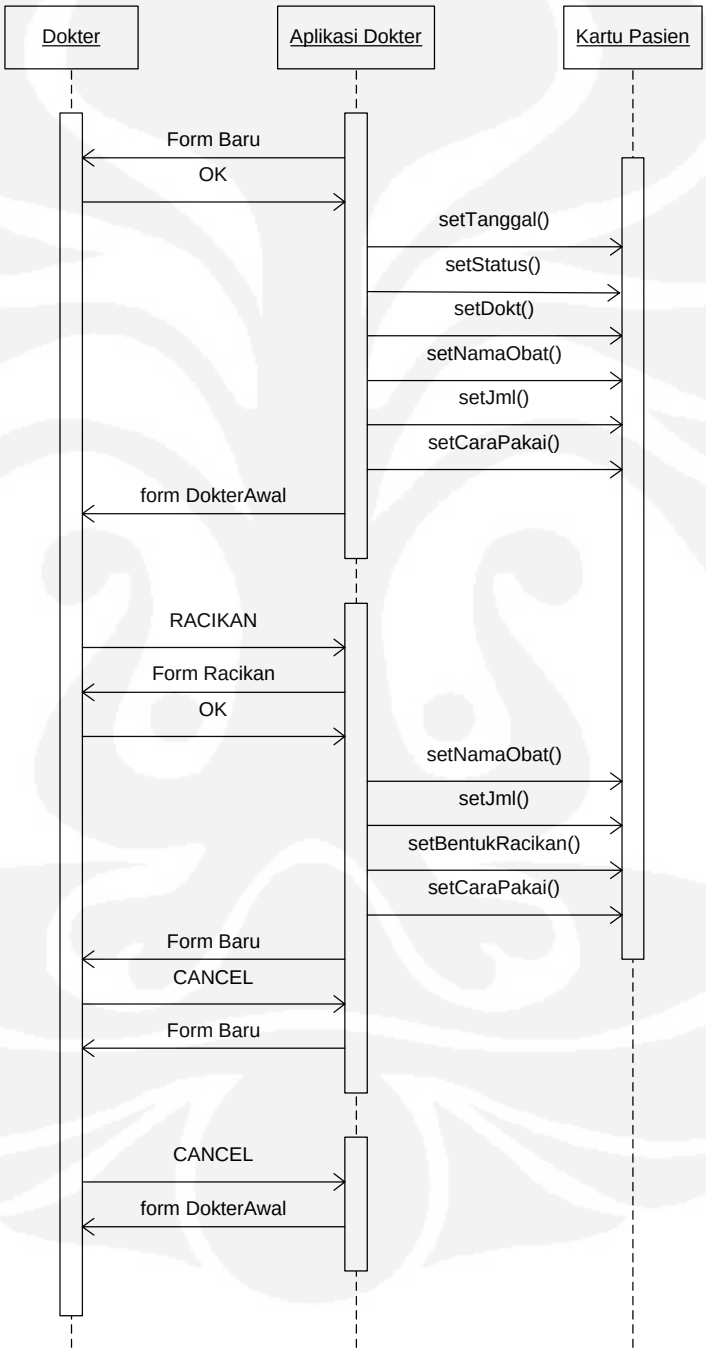
Jika tidak sama, maka akan kembali ke tampilan awal yang menampilkan pilihan untuk melihat resep lama atau membuat resep baru. Jika sama maka akan terlihat obat apa saja yang terdapat pada resep tersebut, dan tanggal berapa resep itu dibuat.



Gambar 3.6 Diagram sekuensial aplikasi dokter resep lama

Jika dokter memilih membuat resep baru maka dokter diharuskan memasukkan tanggal, obat apa saja yang harus diambil oleh pasien, dan cara pakai obat tersebut. Status resep dapat di tulis karena pada beberapa kasus resep tidak dapat di ulang atau harus segera ditebus karena sangat mendesak.

Pada aplikasi dokter ini terdapat tombol racikan yang akan mengeluarkan tampilan baru yang dapat diisi oleh dokter obat apa saja yang ingin dijadikan racikan dan ingin disajikan dalam bentuk apa obat tersebut, apakah akan menjadi puyer, dimasukkan dalam kapsul ataupun dalam bentuk sirup.

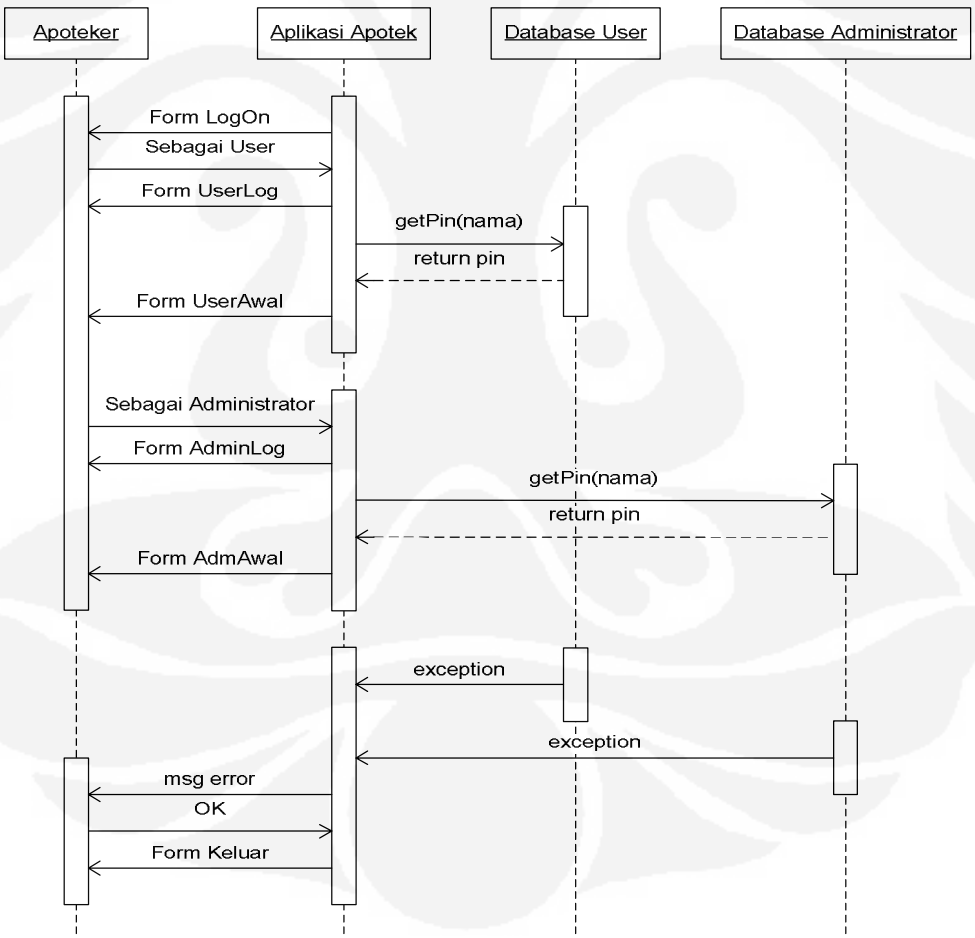


Gambar 3.7 Diagram sekuensial aplikasi dokter resep baru

Pada diagram sekuensial aplikasi login apotek seperti yang di tunjukkan gambar 3.8, merupakan pemisah dari pengguna dalam apotek. Pada saat pengguna memilih sebagai user, maka aplikasi apotek akan mengeluarkan form yang digunakan untuk pengisian nama dan pin. Jika namanya terdapat pada database dan pinnya cocok maka program user akan berlangsung. Dan jika tidak maka akan tampil.

Begitu juga jika ingin masuk sebagai administrator. Setelah pengguna memasukkan nama dan pin. Maka program aplikasi ini akan mencari nama tersebut pada database dan mengambil pin yang terdapat pada database tersebut dan mencocokannya dengan pin yang di tulis oleh pengguna.

Pada aplikasi ini pengguna di beri kesempatan tiga kali pada form yang sama, jika pada saat ketiga pin tetap salah, maka akan keluar dari aplikasi ini.



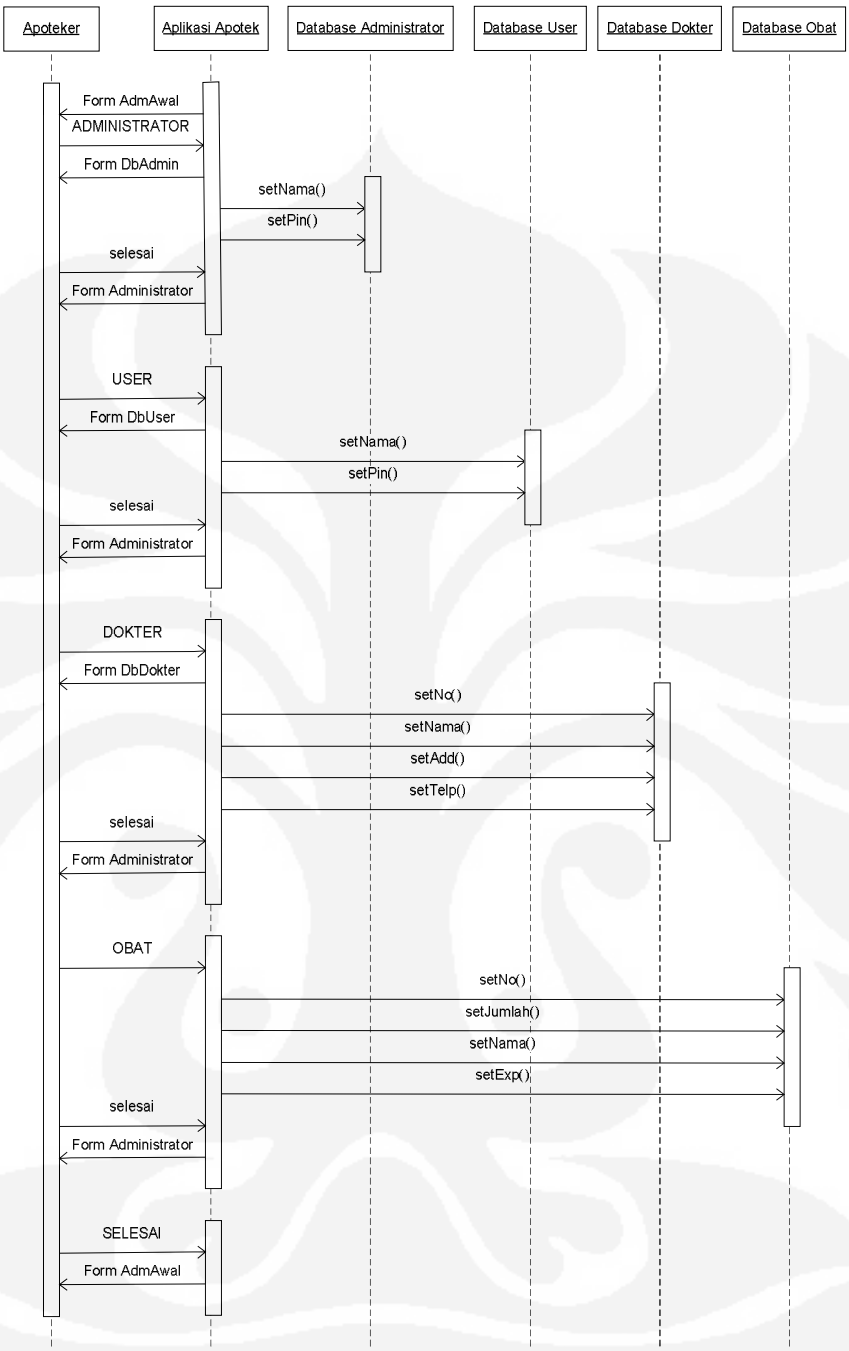
Gambar 3.8 Diagram sekuensial aplikasi login apotek

Aplikasi administrator apotek ini, digunakan untuk mengedit sistem database. Biasanya digunakan pada saat ada kesalahan pada database maka akan dapat di ubah. Terdapat empat pilihan yang ingin di edit yaitu administrator, user, obat dan dokter.

Bila kita memilih aplikasi administrator atau pun aplikasi user, kita dapat menambahkan, menghapus, maupun mengedit pengguna yang nantinya dapat mengakses sesuai dengan bagiannya.

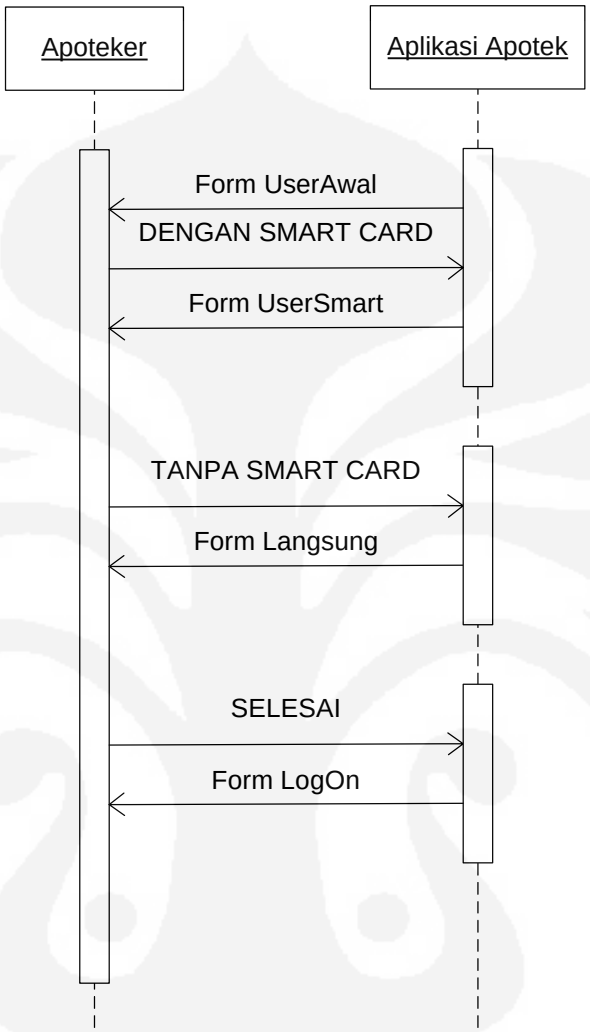
Pada pilihan obat kita dapat mengedit nomor urut obat, jumlah obat, nama obat maupun *expired date* dari obat tersebut. Pada aplikasi ini, kita dapat melihat obat –obat dalam bentuk tabel yang dapat di urutkan seuai keinginan kita untuk mempermudah pemilihan obat yang ingin diedit. Biasanya digunakan jika terdapat obat yang telah mendekati masa *expired* ataupun jika terdapat kesalahan pada jumlah obat.

Untuk aplikasi pengeditan dokter kita akan dapat mengedit kode dokter, alamat dokter, dan nomor telp dokter tersebut jika terjadi perubahan. Aplikasi ini juga ditampilkan dalam bentuk tabel untuk mempermudah pencarian dokter



Gambar 3.9 Diagram sekuensial aplikasi edit administrator apotek

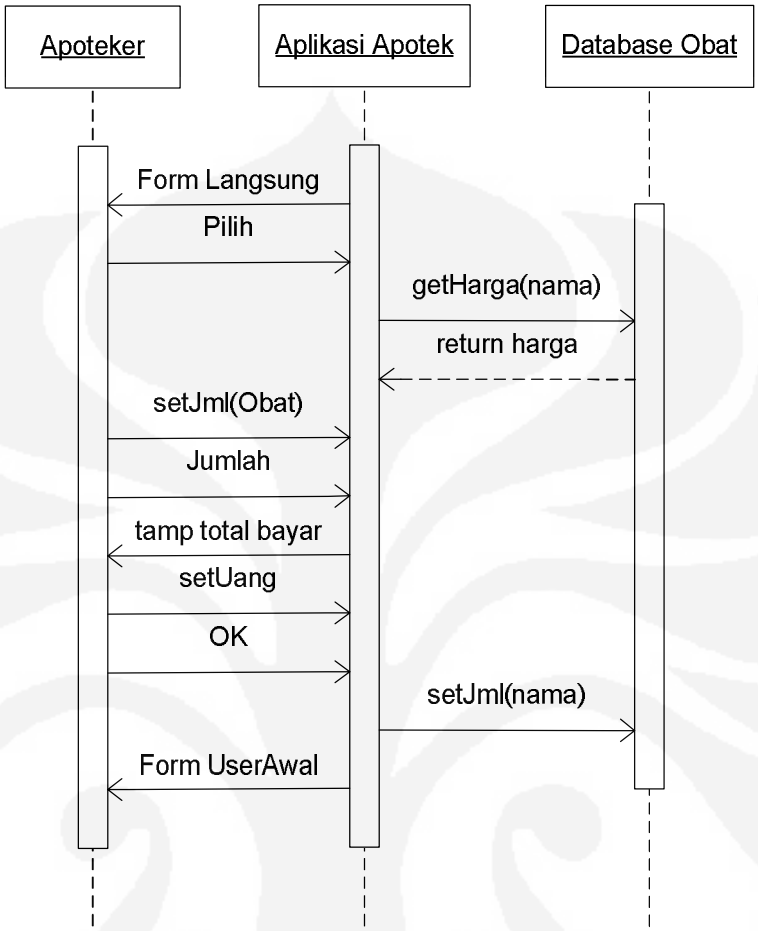
Setelah masuk pada aplikasi user maka akan ada dua pilihan yaitu dengan *smart card* atau tanpa *smart card*. Dengan *smart card* digunakan jika pelanggan tersebut menggunakan *smart card* sebagai resepnya sedangkan tanpa *smart card* berfungsi untuk melayani pelanggan yang tidak menggunakan resep atau menggunakan resep tetapi bukan dalam bentuk *smart card* melainkan kertas.



Gambar 3.10 Diagram sekuensial aplikasi user awal

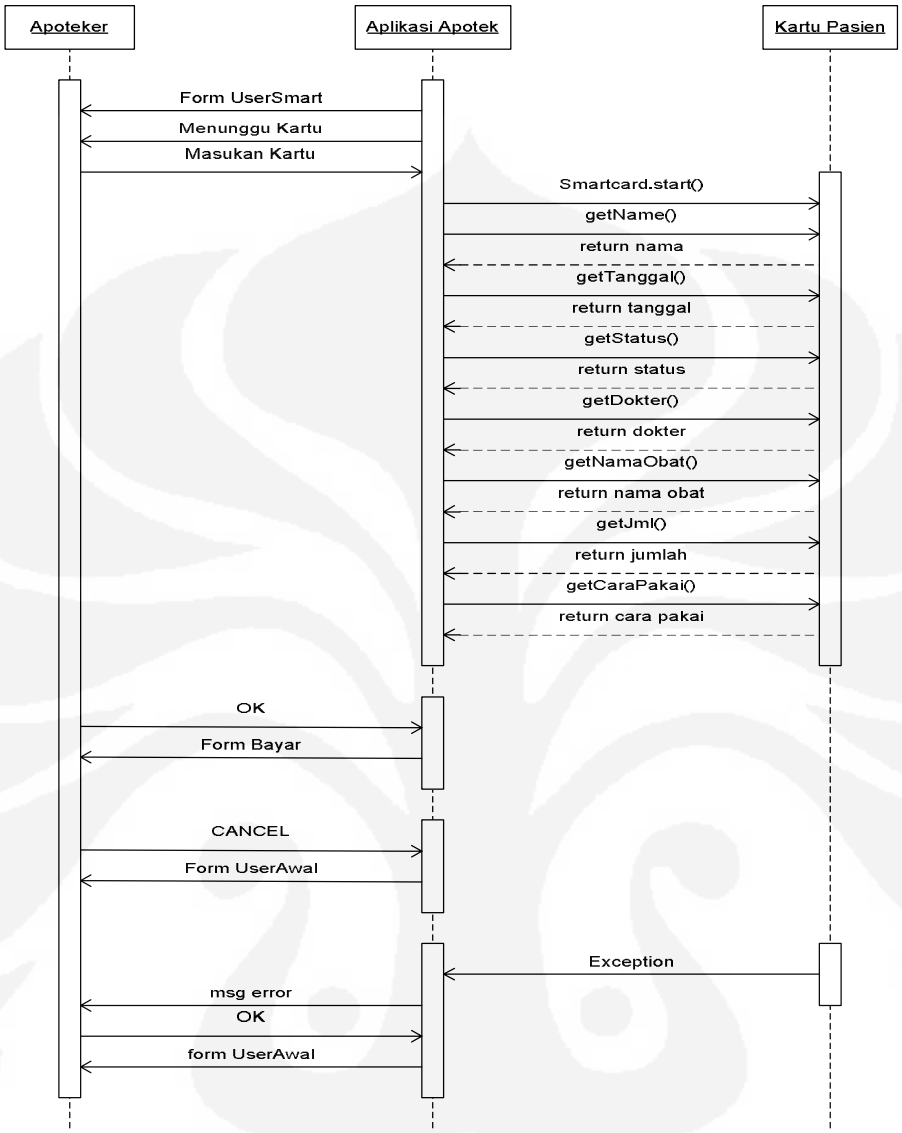
Aplikasi user pembalian langsung adalah aplikasi yang di gunakan untuk melayani pelanggan yang tidak menggunakan *smart card* sebagai resepnya. Pada aplikasi ini, apoteker melihat obat yang tesedia dalam bentuk tabel dan dapat memilih obat yang ingin dibeli. User memasukkan jumlah obat yang ingin dibeli, maka akan tertera jumlah yang harus dibayar oleh pelanggan.

Pada saat uang telah dibayar, maka jumlah obat pada database akan berkurang sesuai jumlah yang dibeli oleh pelanggan tersebut. Diagram sekuensialnya dapat dilihat pada gambar dibawah ini:



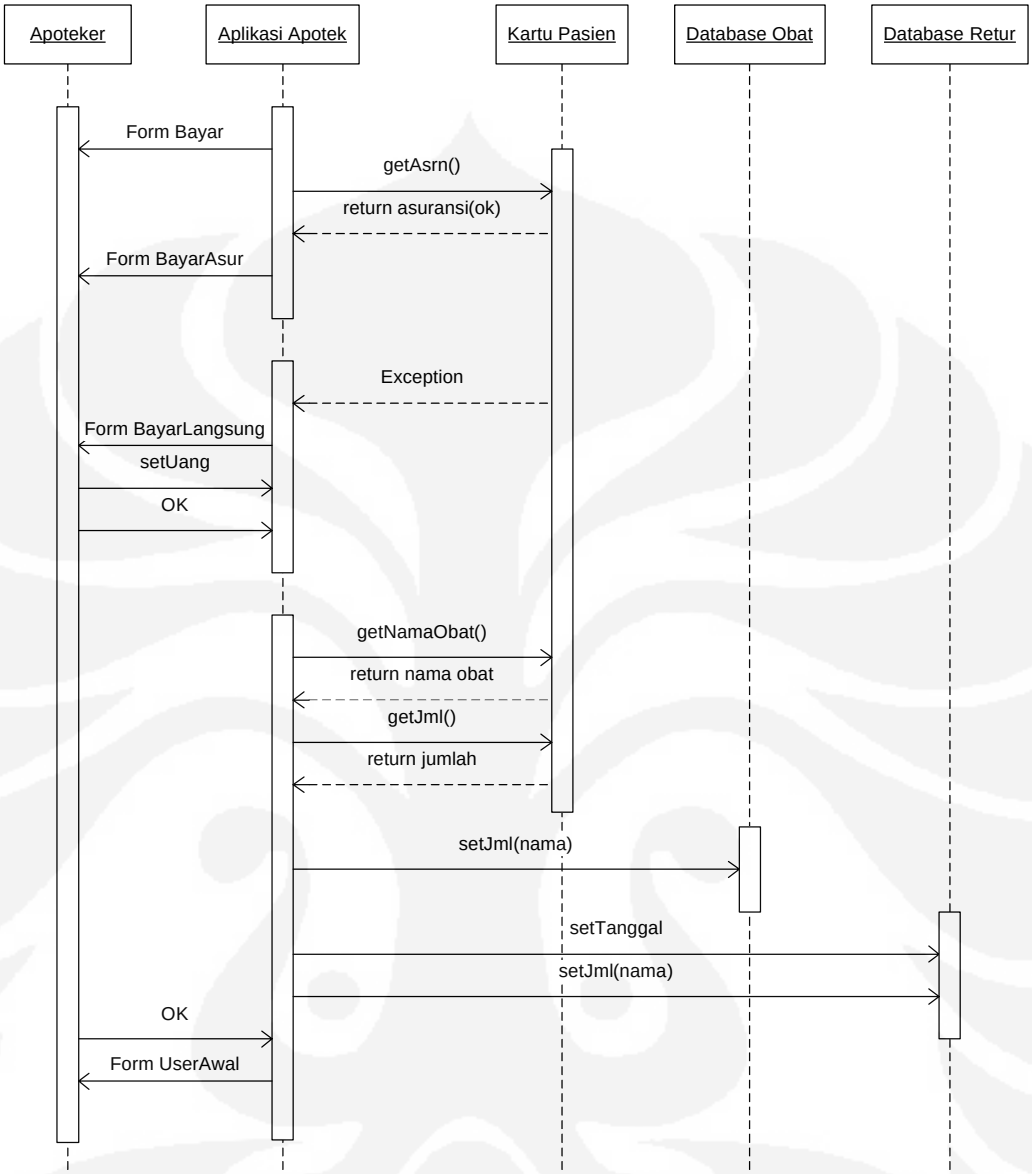
Gambar 3.11 Diagram sekuensial aplikasi user pembelian langsung

Untuk pembelian yang menggunakan *smart card*, akan muncul form yang mengisyaratkan untuk memasukkan kartu dan aplikasi apotek akan menunggu kartu. Ketika kartu dimasukkan maka akan mengaktifkan power kartu dengan pesan *Smartcard.start()*. Aplikasi apotek akan membaca resep yang terdapat pada *smart card* tersebut yang kemudian ditampilkan. Ketika user menekan tombol OK, maka akan berlanjut pada form pembayaran. Diagram sekuensial aplikasi pembelian dengan menggunakan *smart card* ditunjukkan pada gambar 3.12.

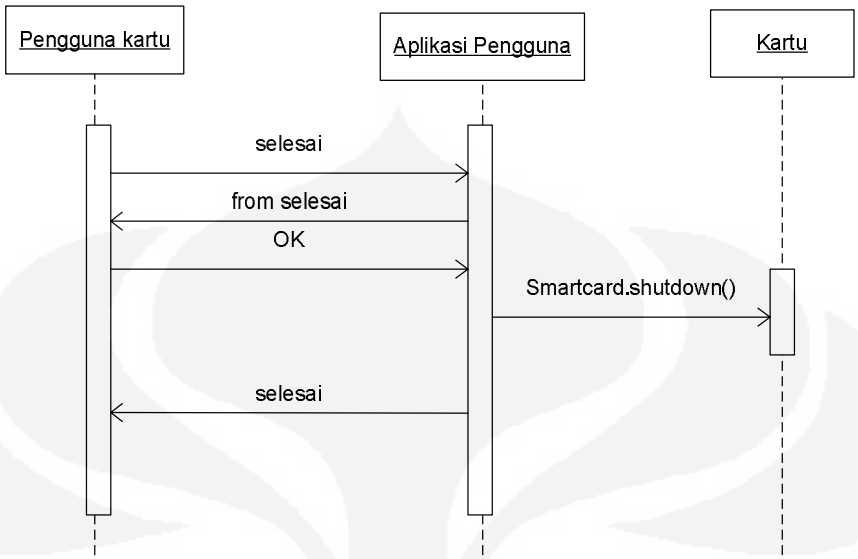


Gambar 3.12 Diagram sekuensial aplikasi user pembelian dengan menggunakan *smart card*

Pada aplikasi pembayaran ini hanya di akses dengan menggunakan *smart card*. Aplikasi ini membedakan pelanggan yang menggunakan asuransi dengan yang tidak. Jika pada pembacaan *smart card* terdapat asuransi, maka pelanggan tidak perlu mengeluarkan uang karena akan ditanggung oleh pihak asuransi. Jika tidak ada asuransi, maka pelanggan harus tetap mengeluarkan uang untuk pembelian obat tersebut. Jumlah obat pada database akan berkurang, setelah pelanggan membayar obat yang dibelinya.



Gambar 3.13 Diagram sekuensial aplikasi pembayaran



Gambar 3.14 Diagram sekuensial aplikasi keluar

Gambar di atas merupakan proses saat pengguna selesai melakukan transaksi. Kemudian pada saat pengguna menekan tombol selesai, aplikasi user akan mematikan *power* kartu dengan memberikan pesan *Smartcard.shutdown()*. Kemudian menutup aplikasi yang sedang berlangsung.

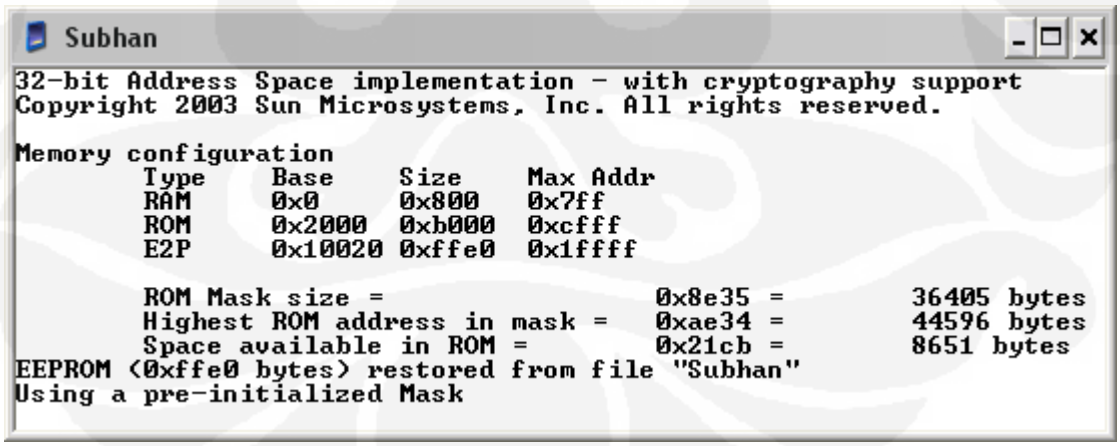
BAB IV
PENGUJIAN DAN ANALISA PROGRAM

4.2 PENGUJIAN PROGRAM

Pengujian program ini bertujuan untuk mengetahui kinerja dari aplikasi yang berkaitan dengan sistem apotek. Aplikasi apotek ini dibagi menjadi dua yaitu aplikasi user dan administrator apotek.

Smart card menggunakan simulator *C-languge Java Card Runtime Environment (C-JCRE)* yang digunakan untuk mensimulasikan *persistant memory* (EEPROM), menyimpan dan memakai kembali isi dari EEPROM dan *file disk*, sama seperti yang terjadi pada *smart card* yang sesungguhnya dalam hal menggunakan isi dari EEPROM. *Applet* dapat di install pada C-JCRE.

Gambar 4.1 merupakan tampilan pada saat kartu pasien dimasukkan pada aplikasi. *Java Card reference implementation* versi 2.2.1 menyediakan versi 32-bit 64 KB akses memori dari *C-language Java Card RE* dijalankan pada *Microsoft Windows NT platform*. Sedangkan *database* apotek disimpan pada *database* MySQL Control Center 0.9.2-beta-[Console Manager].



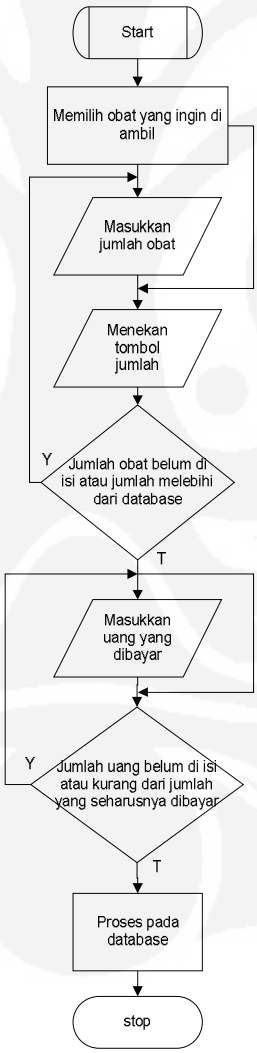
Gambar 4.1 *C-languge Java Card Runtime Environment*

4.2.1 Aplikasi User Apotek

Pengujian aplikasi user apotek digunakan oleh kasir pada apotek. Aplikasi ini dibagi menjadi dua bagian yaitu untuk pembelian tanpa menggunakan *smart card* atau pembelian secara langsung dan pembelian yang menggunakan *smart card*.

4.2.1.1 Pembelian Langsung

Pembelian langsung merupakan aplikasi yang dijalankan jika ada pembelian tanpa resep atau pembelian yang menggunakan resep konvensional yang terbuat dari kertas lihat Gambar 4.2. Pada pembelian langsung ini pasien atau pembeli obat diharuskan membayar obat yang dibelinya. Karena pada aplikasi ini tidak dapat mendeteksi bahwa pembeli tersebut mempunyai asuransi atau tidak.



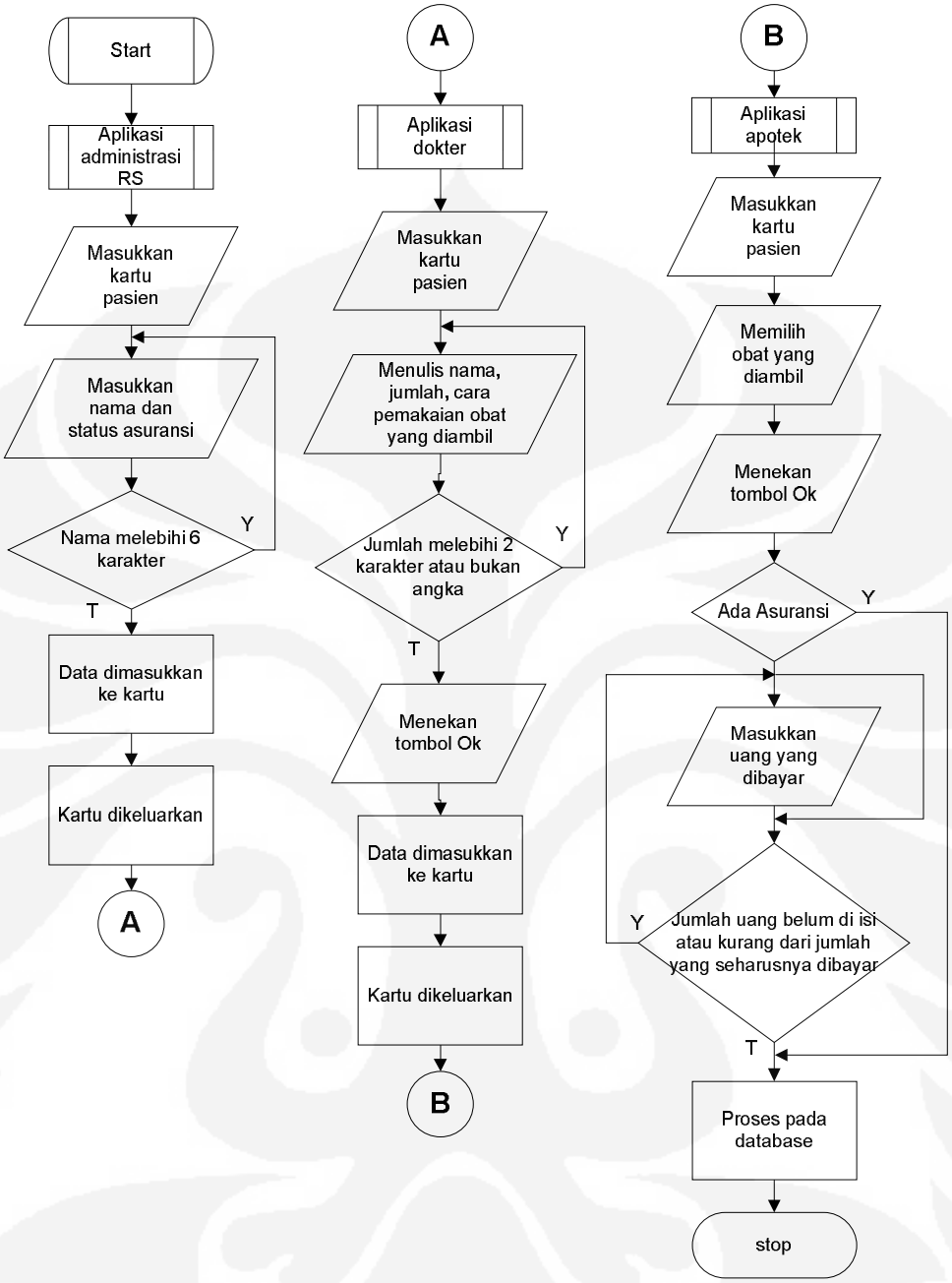
Gambar 4.2 Diagram alir pengujian pembelian langsung.

Pengujian dilakukan dengan membuat suatu skenario cerita :

1. Seorang kasir apotek baru datang dan mencoba melakukan login pada aplikasi.
2. Kasir tersebut lupa nomor pinnya dan melakukan tiga kesalahan percobaan PIN.
3. Kasir tersebut mencoba masuk lagi dan kali ini benar nomor pin nya sehingga dapat masuk ke aplikasi user apotek.
4. Ada pembeli datang dengan membawa kertas resep.
5. Obat yang dibelinya ada yang melebihi dari stok yang ada.
6. Akhirnya hanya mengambil obat yang ada saja.
7. Pada saat pembayaran kasir salah melakukan pembacaan yang harus dibayar, sehingga uang yang dikeluarkan oleh pembeli untuk menebus obat tersebut kurang.
8. Akhirnya dia meralat perkataannya, dan pembeli tersebut menambah uangnya.
9. Kasir memberikan kembalian dan obat yang dibeli kepada pembeli tersebut.

4.2.1.2 Pembelian Dengan Smart Card

Pengujian dengan *Smart Card* dimulai dari kartu diinisialisasi pada aplikasi administrasi rumah sakit, lalu diisi dengan resep dokter setelah itu baru dapat dilakukan pembelian obat pada apotek. Gambar 4.3 menunjukkan alur dari pengujian terhadap aplikasi apotek dengan menggunakan *smart card*.



Gambar 4.3 Diagram alir pengujian pembelian dengan *smart card.*

Pengujian ini dilakukan dengan membuat suatu skenario cerita :

1. Suatu rumah sakit mempunyai fasilitas *smart card* sebagai kartu pasiennya.
2. Seorang pasien baru melakukan pendaftaran pada administrasi RS.
3. Seorang petugas rumah sakit membuatkan kartu baru dengan diisikan nama pasien tersebut dan status asuransi pasien tersebut.

4. Pasien tersebut berobat ke dokter Budiman lalu diberikan resep oleh dokter tersebut.
5. Dokter tersebut merasa ada obat yang lupa ditulis sehingga dokter tersebut memasukkan kembali kartu pasien tersebut dan melihat resep yang telah di tulisnya. Ternyata sudah benar sehingga tidak perlu menulis ulang obat tersebut.
6. Pasien tersebut ke apotek untuk menebus obat.
7. Karena pasien menggunakan asuransi yang bekerja sama dengan apotek maka pasien tersebut tidak perlu membayar atas obat yang ditebusnya.
8. Ada pembeli obat yang lain datang dengan membawa *smart card* dengan obat yang tidak semuanya diambil karena sudah di ambil di apotek yang lain.
9. Kasir memilih obat mana saja yang harus di ambil oleh pasien ini.
10. Obat di bayar.
11. Uji coba selesai, kartu di keluarkan.

4.2.2 Aplikasi Administrator Apotek

Aplikasi administrator apotek digunakan untuk mengedit empat hal yakni administrator, user, dokter,dan obat.

Pengujian pada administrator dapat dilihat jika pegawai administrator ditambah dengan memasukkan nama pegawai tersebut dan pin. Pin tersebut digunakan untuk masuk pada aplikasi administrator. Pengujian dilakukan dengan keluar dari aplikasi dan masuk lagi pada aplikasi administrator menggunakan nama orang tersebut.

Begitu juga dengan pengeditan aplikasi user, pada saat ada pegawai yang baru maka nama dan pin pegawai tersebut akan dimasukkan pada sistem database agar dapat mengakses pada aplikasi user apotek. Setelah itu, akan di coba untuk masuk pada aplikasi user.

Untuk pengeditan dokter digunakan jika ada dokter baru, atau ada dokter yang pindah alamat atau berganti nomor telepon. Pengujian dilakukan dengan

menambah dokter baru dan dicoba dengan menggunakan *smart card* dengan cara penulisan resep baru dengan id dokter tersebut.

Pengeditan obat pada aplikasi ini dilakukan jika ada obat yang sudah habis masa berlakunya maka akan dihapus, atau terdapat penambahan obat baru. Pengujian dilakukan dengan cara mengisi obat baru lalu kita melihat pada aplikasi user apotek dengan pembelian secara langsung.

4.2.3 Pengujian Oleh Responden

Pengujian dilakukan oleh dua puluh orang responden yang terdiri dari berbagai profesi seperti dokter, apoteker, pegawai apotek dan mahasiswa kedokteran. Parameter penilaian setiap aplikasi didasarkan pada beberapa parameter penilaian yang meliputi efisiensi waktu pada apotek, kelengkapan informasi, pemahaman terhadap program aplikasi yang dibuat, perlunya komputerisasi pada apotek.

Setelah dilakukan pengujian (survei), dilakukan analisa dengan hipotesa perhitungan untuk menilai 95% *confidence interval* (rentang kepercayaan).

$$stdeviasi = \frac{\sqrt{\sum(nilai - mean)^2}}{populasi - 1}$$

$$mean \pm \left(1{,}96 \cdot \frac{stdeviasi}{\sqrt{populasi}}\right)$$

Stdeviasi adalah standar deviasi yang berguna untuk menghitung besar penyimpangan (error) di sekitar nilai rata-rata (mean). Sedangkan populasi adalah jumlah banyak responden. Mean (rata-rata) diambilkan dari nilai rata-rata yang didapatkan dari hasil survei.

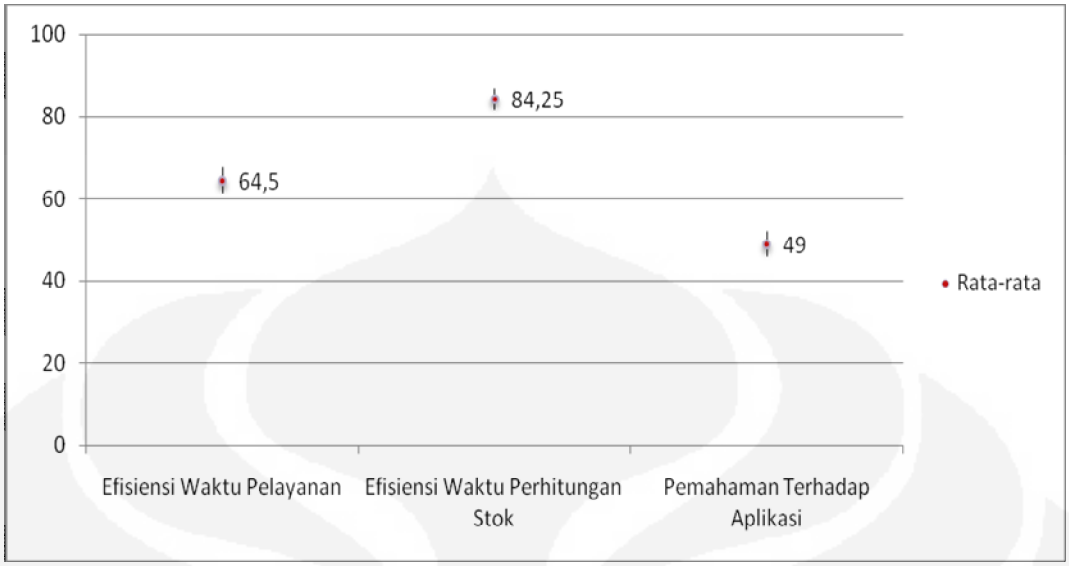
4.2.3.1 Aplikasi apotek tanpa menggunakan smart card

Setelah diadakan pengujian sesuai parameter yang diajukan, maka didapatlah Tabel 4.1.

Tabel 4.1 Tabel Pengujian Responden pada Aplikasi Apotek Tanpa Smart Card

Parameter Penilaian / Responden	Efisiensi Waktu Pelayanan	Efisiensi Waktu Perhitungan Stok	Pemahaman Terhadap Aplikasi
Responden 1	65 %	65 %	35 %
Responden 2	65 %	100 %	100 %
Responden 3	65 %	65 %	35 %
Responden 4	100 %	100 %	35 %
Responden 5	65 %	100 %	65 %
Responden 6	35 %	65 %	35 %
Responden 7	65 %	100 %	100 %
Responden 8	35 %	65 %	35 %
Responden 9	100 %	100 %	35 %
Responden 10	35 %	100 %	65 %
Responden 11	65 %	65 %	35 %
Responden 12	35 %	100 %	65 %
Responden 13	65 %	65 %	35 %
Responden 14	100 %	100 %	35 %
Responden 15	65 %	100 %	35 %
Responden 16	65 %	65 %	35 %
Responden 17	35 %	100 %	65 %
Responden 18	65 %	65 %	35 %
Responden 19	100 %	100 %	35 %
Responden 20	65 %	65 %	65 %
Rata-rata	64,5 %	84,25 %	49 %
Standar deviasi	5,101464	4,098425	5,001385
Batas atas	67,66192015	86,79023013	52,09989054
Batas bawah	61,3380779	81,7097699	45,9001095

Melihat data pada Tabel 4.1 maka kita dapat mengambil rata-rata, batas atas dan batas bawah sehingga dapat dibuatlah grafik seperti pada Gambar 4.4.



Gambar 4.4 Grafik hasil prosentase pengujian oleh responden pada aplikasi apotek tanpa *smart card.*

Pada grafik waktu pelayanan kita melihat prosentase aplikasi dapat meningkatkan kecepatan pelayanan yang tidak menggunakan *smart card* dengan menggunakan program sebesar 64,5 %, karena lebih cepat dibandingkan dengan perhitungan manual karena pada aplikasi tersebut kita dapat melihat jumlah yang harus dibayar. Menurut prosentase pengujian oleh responden peningkatan efisiensi dari waktu perhitungan stok obat sebesar 84,25 %, karena kita dapat secara langsung mengurangi stok obat yang terdapat pada tempat penyimpanan.

Sedangkan prosentase responden yang memahami aplikasi apotek yang dibuat adalah sebesar 49 %. Beberapa hal yang menjadi alasan responden meliputi kurang terbiasa dengan sistem yang baru, kurang sosialisasi, dan butuh waktu untuk memahami program aplikasi yang baru.

Pada Gambar 4.4 diketahui besar penyimpangan tidak terlalu besar karena penyimpangan yang terjadi bernilai dibawah 5,2 dalam range 100.

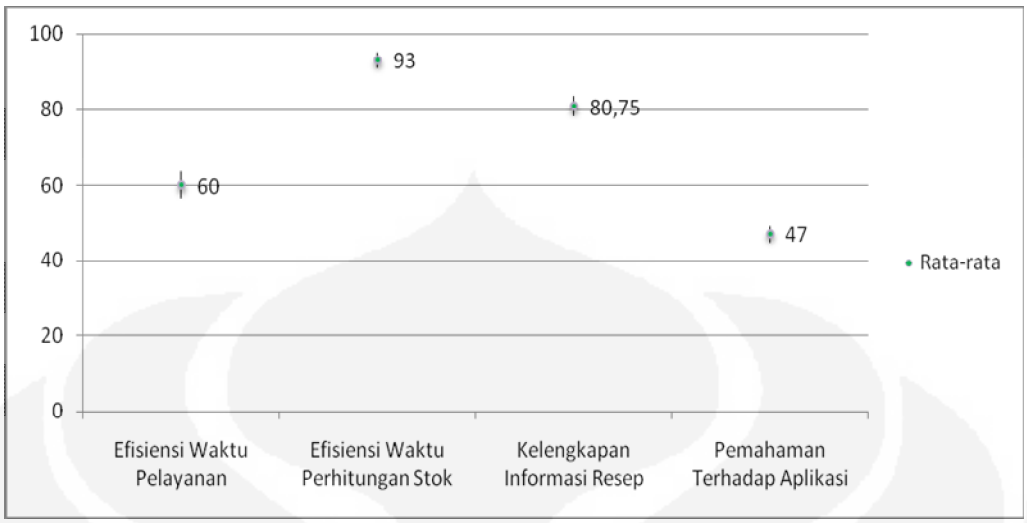
4.2.3.2 Aplikasi apotek dengan menggunakan smart card

Pengujian dilakukan oleh responden yang sama dengan pada aplikasi tanpa *smart card.* Setelah diadakan pengujian sesuai parameter yang diajukan maka didapatlah Tabel 4.2.

Tabel 4.2 Tabel Pengujian Responden pada Aplikasi Apotek dengan *Smart Card*

Parameter Penilaian / Responden	Efisiensi Waktu Pelayanan	Efisiensi Waktu Perhitungan Stok	Kelengkapan Informasi Resep	Pemahaman Terhadap Aplikasi
Responden 1	35 %	100 %	100 %	35 %
Responden 2	65 %	100 %	65 %	65 %
Responden 3	35 %	65 %	100 %	35 %
Responden 4	65 %	100 %	65 %	35 %
Responden 5	65 %	100 %	65 %	65 %
Responden 6	35 %	100 %	100 %	35 %
Responden 7	35 %	100 %	65 %	65 %
Responden 8	100 %	65 %	100 %	35 %
Responden 9	65 %	100 %	65 %	35 %
Responden 10	65 %	100 %	65 %	65 %
Responden 11	35 %	100 %	100 %	35 %
Responden 12	100 %	100 %	65 %	65 %
Responden 13	35 %	65 %	100 %	35 %
Responden 14	65 %	100 %	65 %	35 %
Responden 15	65 %	100 %	65 %	65 %
Responden 16	35 %	100 %	100 %	35 %
Responden 17	100 %	100 %	65 %	65 %
Responden 18	35 %	65 %	100 %	35 %
Responden 19	65 %	100 %	65 %	35 %
Responden 20	100 %	100 %	100 %	65 %
Rata-rata	60 %	93 %	80,75 %	47 %
Standar deviasi	5,668595	3,295258	4,098425	3,4593
Batas atas	63,51343158	95,04242207	83,29023013	49,14409635
Batas bawah	56,4865684	90,9575779	78,2097699	44,8559036

Gambar 4.5 merupakan grafik tanggapan pengguna yang menyimpulkan nilai rata-rata pengujian dan besar penyimpangan dari hasil yang didapat pada Tabel 4.2.



Gambar 4.5 Grafik hasil prosentase pengujian oleh responden pada aplikasi apotek dengan *smart card.*

Pada grafik hasil pengujian kita melihat bahwa kecepatan pelayanan dengan menggunakan aplikasi apotek yang menggunakan *smart card* sebesar 60 %. Hal yang menjadi alasan adalah jika pasien tidak ingin mengambil semua dari jumlah obatnya, sang apoteker juga harus memperhitungkan dosis dari si pasien diluar program setelah itu baru menulis lagi pada aplikasi sehingga pekerjaan mereka menjadi dua kali dari biasanya.

Prosentase responden yang menyetujui bahwa program aplikasi dengan *smart card* dapat menambah efisiensi waktu melihat hasil perhitungan stok adalah sebesar 93 %. Karena dapat terlihat obat apa saja yang tersedia dan yang tidak tersedia.

Untuk kelengkapan informasi resep diperoleh prosentase sebesar 80,75 % karena untuk resep yang didalamnya terkandung zat narkotika haruslah ditulis alamat si pasien dan pada aplikasi ini belum terdapat alamat pasien.

Sedangkan prosentase responden yang memahami aplikasi tersebut adalah sebesar 47 %. Beberapa hal yang menjadi alasan responden meliputi kurang terbiasa dengan sistem yang baru, kurang sosialisasi, dan butuh waktu untuk memahami program aplikasi yang baru.

Pada Gambar 4.5 diketahui besar penyimpangan tidak terlalu besar karena bernilai kurang dari 6 persen dalam range 100.

4.3 ANALISA HASIL PENGUJIAN PROGRAM

Setelah program diujicobakan maka diketahui bahwa aplikasi tersebut masih dapat dikembangkan agar dapat mencakup semua fungsi yang diperlukan pada apotek. Hasil yang didapat yaitu:

1. Kartu harus masuk secara berurutan dari administrasi rumah sakit, dokter lalu apotek agar dapat menjalankan semua aplikasi.
2. Apabila pengguna mencoba melakukan aplikasi yang menggunakan *smart card* tetapi tidak memasukkan kartu (menjalankan C-JCRE) maka program akan memberikan pesan bahwa kartu belum dimasukkan.
3. Panjang maksimal untuk nama pasien yang dimasukkan oleh pegawai administrasi rumah sakit pada kartu adalah 8 karakter.
4. Panjang maksimal untuk alamat pasien yang dimasukkan oleh pegawai administrasi rumah sakit pada kartu adalah 30 karakter.
5. Dokter dapat melihat resep yang telah ditulis sebelumnya, jika pasien merupakan pasien yang baru maka resep yang ditulis sebelumnya kosong.
6. Jumlah obat yang dimasukkan oleh dokter harus menggunakan angka. karena jika tidak maka akan ada kesalahan pada waktu disimpan di kartu yang berakibat pembacaan pada apotek menjadi ”null” atau tidak dapat dibaca.
7. Jumlah obat yang dimasukkan maksimal dua karakter, jika tetap dicoba menulis dengan menggunakan 3 karakter maka pada pembacaan *smart card* pada apotek tidak dapat dibaca.
8. Obat yang ditulis oleh dokter harus sesuai dengan yang terdapat pada database, tetapi dapat menggunakan huruf besar atau kecil maupun kombinasinya.
9. Keberagaman obat maksimum yang dapat digunakan adalah 4 macam.
10. Apabila pengguna apotek mencoba melakukan login dan melakukan kesalahan sampai tiga kali maka akan keluar dari program tersebut.
11. Identitas dari dokter yang menulis apotek dapat terlihat pada aplikasi pembelian obat yang mengunakan *smart card*.

12. Apabila pada saat pembayaran uang kurang dari yang harus dibayarkan maka program akan memberikan pesan bahwa uang yang dibayarkan kurang.
13. Pada penggunaan *smart card* pada apotek kita dapat langsung mengetahui obat apa saja yang akan ditebus dan akan tertera jika obat tersebut tersedia atau tidak.
14. Untuk obat racikan jika salah satu obat saja tidak tersedia maka obat racikan tidak dapat diambil karena racikan akan tidak sesuai dengan yang dibutuhkan.
15. Kartu akan dikeluarkan (C-JCRE tertutup) secara otomatis jika proses telah selesai.
16. Pasien yang menggunakan asuransi hanya berbeda pada saat pembayaran.
17. Pegawai apotek dapat memilih obat apa saja yang ingin diambil oleh pasien pada pembelian dengan *smart card* untuk pembelian obat yang hanya sebagian.
18. Apoteker harus menghitung kembali dosis obat yang akan diambil oleh pasien jika pasien tersebut hanya ingin mengambil setengah dari obat yang dibutuhkan. Walaupun sebenarnya hal tersebut tidak diperkenankan karena jika sang pasien hanya mengambil setengahnya maka ada kemungkinan virus yang terdapat pada si pasien masih belum hilang sepenuhnya sehingga virus tersebut akan mencoba mempertahankan diri sehingga dosis obat harus ditambah untuk pengobatan selanjutnya.
19. Penyimpangan yang didapat pada saat survey tidak terlalu besar karena bernilai dibawah 6 dalam range 100.
20. Prosentase responden yang memahami aplikasi masih di bawah 50 % sehingga adanya sosialisasi dan pelatihan terhadap aplikasi sangat diperlukan agar optimal penggunaannya.

BAB V
KESIMPULAN

Setelah dilakukan analisa dan ujicoba maka dapat diambil suatu kesimpulan sebagai berikut:

1. Pada aplikasi apotek merupakan aplikasi yang *multi user* dimana tiap *user* mempunyai otoritas masing-masing sehingga mengurangi terjadinya kesalahan dalam pemasukan data. Selain itu aplikasi ini dapat digunakan untuk pembelian dengan menggunakan *smart card* dan tidak menggunakan *smart card.*
2. Perhitungan transaksi yang penggunaan *smart card* diproses secara otomatis dan terkomputerisasi sehingga dapat mempercepat dan menghindari terjadinya penyimpangan data. Jika operator melakukan kesalahan dalam menggunakan aplikasi, maka program akan memberikan pesan kesalahan.
3. Pengembangan yang dapat dilakukan terhadap aplikasi ini mencakup:
    a. Pengaplikasian pada smart card yang sesungguhnya.
    b. Pencarian obat yang sejenis, jika obat yang dinginkan tidak tersedia.
    c. Perhitungan dosis obat, jika pasien tersebut hanya mengambil sebagian obat.
4. Dengan melihat analisa data yang menunjukkan rata-rata pada aplikasi apotek tanpa menggunakan *smart card* dan yang menggunakan *smart card* bernilai 49% dan 47% maka perlu adanya sosialisasi dan pelatihan terhadap orang yang berhubungan dengan aplikasi agar dapat digunakan dengan optimal.

DAFTAR ACUAN

[1] Yanfar, *PP RI No.25 Thn. 1980 Tentang Perubahan Atas PP No.26 Thn.1965 Tentang Apotik*, 2004, diakses pada tanggal 15 Maret 2008 http://www.tempointeraktif.com/hg/peraturan/prn,20040412-05,id.html

[2] PP No.25/Tahun 1980

[3] Anief, Mohammad, *Prinsip Umum dan Dasar Farmakologi*, Gajah Mada University Press, 2004

[4] Hendro, *Mengenal Asuransi Kesehatan*, 2007, diakses pada tanggal 21 Maret 2008, http://www.tipsasuransi.blogspot.com/2008/02/mengenal-asuransi-kesehatan.html

[5] *Asuransi perawatan kesehatan*, 2005, diakses pada tanggal 21 Maret 2008, http://www.bringinlife.co.id/kesehatan.aspk.htm

[6] Enrique C. Ortiz, *An Introduction to Java Card Technology*, 2003, diakses pada tanggal 23 Januari 2008, http://www.developers.sun.com/techtopics/mobility/javacard/articles/javacard1/

[7] Adityo Hidayat, *SC dan RFID sayuri*, diakses pada tanggal 21 Mei 2008, www.te.ugm.ac.id/~bimo/download/future_it/tugas/Tugas%20Kelompok%201/**SC**+**RFID**%20**sayuri**.ppt

[8] Java Card™ 2.2, *Application Programming Interface, Revision 1.1 for the 2.2_01 Release*, Sun Microsystems, Inc., 2002.

[9] Ed Ort, *Developing a Java Card Applet*, 2001, diakses pada tanggal 27 Januari 2008, http://www.developers.sun.com/techtopics/mobility/javacard/articles/applet/Wallet.java

[10] Zhiqun Chen, *How To Write A Java Card Applet, A Developer's Guide, Learn the Programming Concepts and Major Steps of Creating Java Card Applets*, 07/01/99, diakses pada tanggal 27 Januari 2008, http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javacard.html?page=1.

[11] *Java Card Applet Developer's Guide*, Sun Microsystems, Inc., Revision 1.12,1998

DAFTAR PUSTAKA

Anief, Mohammad, *Prinsip Umum dan Dasar Farmakologi*, Gajah Mada University Press, 2004

Eckel, Bruce, *Thinking in Java 3rd Edition*, Revision 4, President, MindView, Inc.2002.

Java Card Applet Developer's Guide, Sun Microsystems, Inc., Revision 1.12, 1998

Java Card™ 2.2, *Off-Card Verifier*, Sun Microsystems, Inc., 2002.

Java Card™ Platform, Version 2.2.2, *RMI Client Application Programming Interface*, Sun Microsystems, Inc., Revision 1.12, 2006.

Martin, Hugues , du Bousquet, Lydie, *Automatic Test Generation for Java-Card Applets*

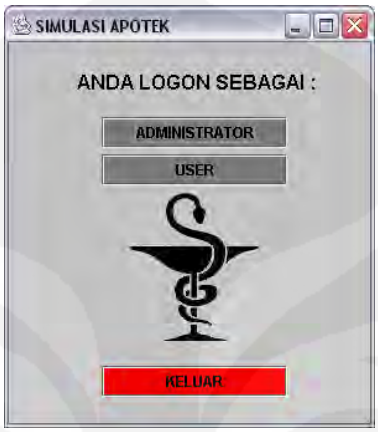
Schildt, Herbert, *The Complete Reference Java™ 2 Forth Edition*, Osborne, 2000.

Supardi, Yuniar, *Pemrograman Database dengan Java dan MySQL*, PT Elex Media Komputindo, 2007.

LAMPIRAN

Lampiran 1 Tampilan Aplikasi Apotek

Di bawah ini merupakan tampilan-tampilan dari aplikasi apotek. Pada Gambar 1 merupakan pemilihan login sebagai *administrator* atau *user*.



Gambar 1 Frame pilihan login

Gambar 2 merupakan frame login yang digunakan untuk membedakan pengguna.

Gambar 2 Frame login

Pada Gambar 3 menunjukkan tampilan untuk memilih apakah pembeli menggunakan *smart card* apa tidak.

Gambar 3 Frame user apotek

Pada Gambar 4 menunjukkan bahwa aplikasi menunggu untuk memasukan kartu. Setelah kartu dimasukkan maka akan keluar seperti pada Gambar 5.

Gambar 4 Frame start aplikasi dengan *smart card*

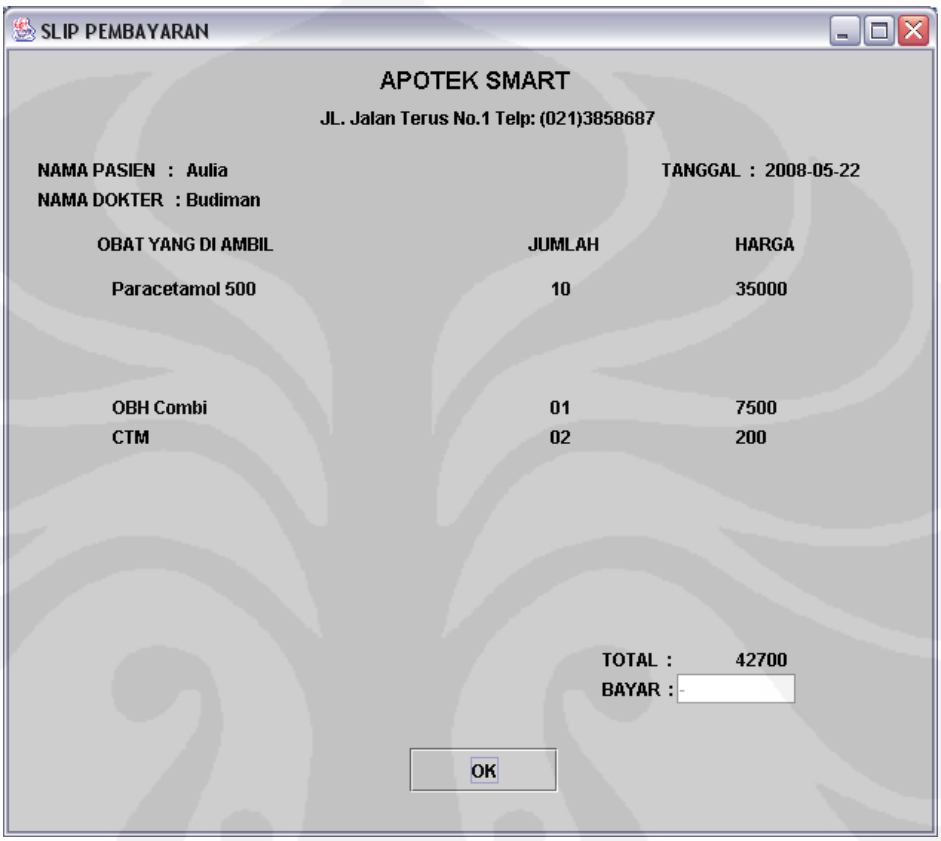
Gambar 5 Frame aplikasi dengan *smart card*

Jika kita menekan ID dokter atau OBAT RACIKAN 1 maka akan muncul tampilan seperti Gambar 6 atau Gambar 7.

Gambar 6 Frame identitas dokter

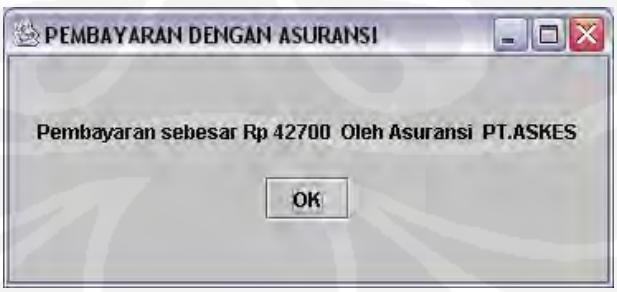
Gambar 7 Frame obat racikan

Setelah memilih obat yang di ambil maka akan tampil Gambar 8. Jika pasien tersebut menggunakan asuransi maka pada kolom bayar akan terisi dengan ”-” yang berarti pasien tersebut tidak membayar atas obat yang dibelinya.



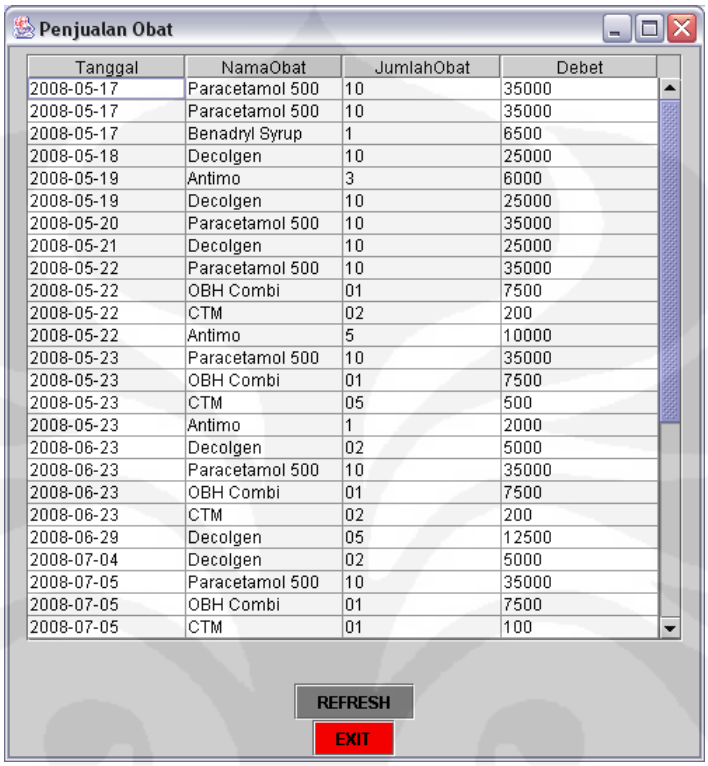
Gambar 8 Frame slip pembayaran

Setelah kita menekan OK maka akan tampli seperti Gambar 9

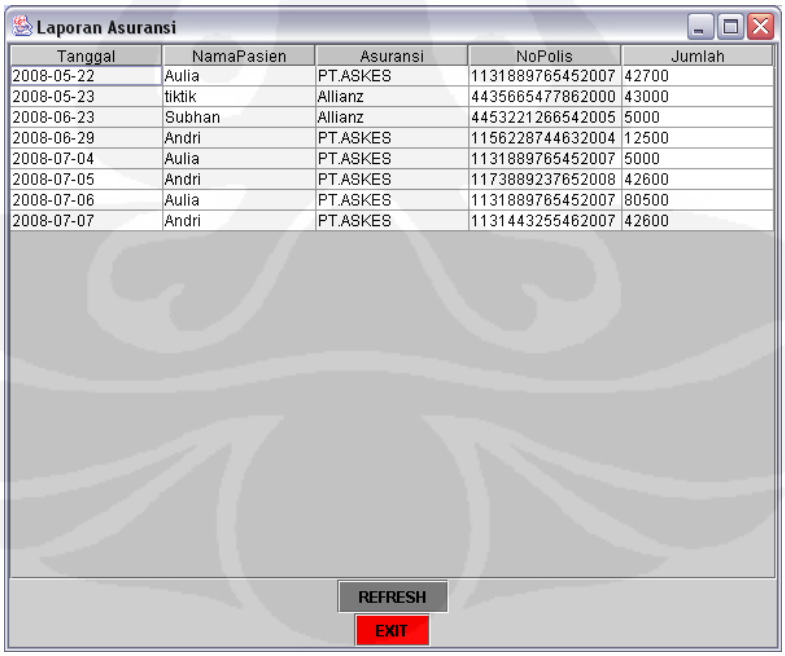


Gambar 9 Frame slip asuransi

Kita dapat melihat barang yang telah dibelinya akan masuk pada tabel penjualan obat pada Gambar 10. Apabila pembayaran menggunakan asuransi maka akan masuk pada tabel asuransi juga lihat Gambar 11.

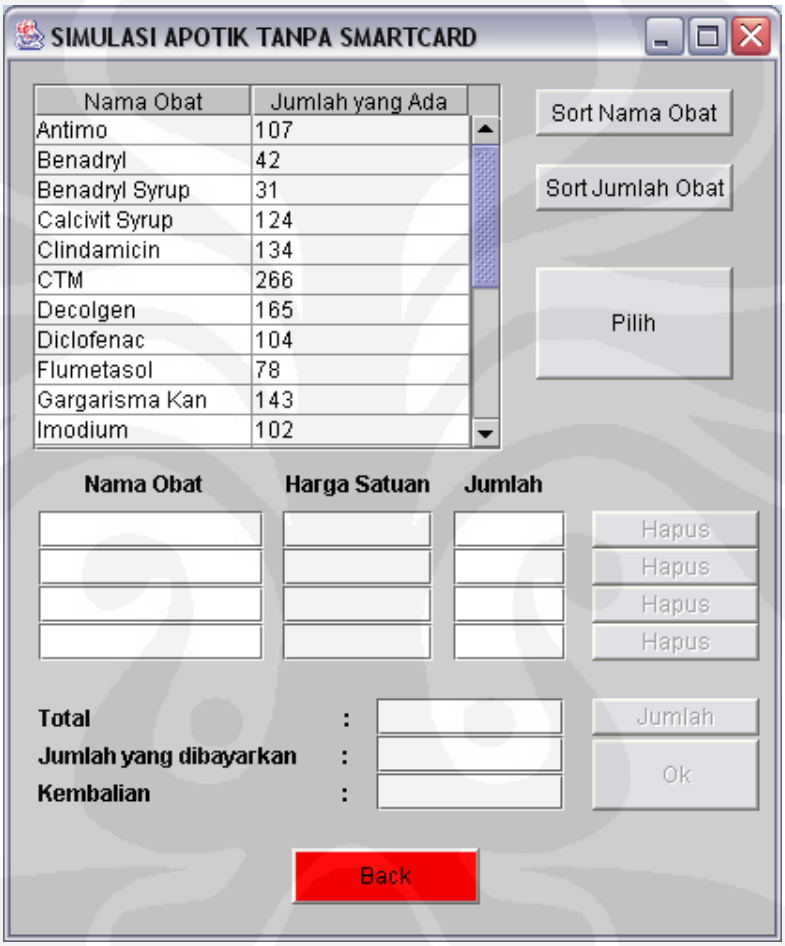


Gambar 10 Frame laporan penjualan.



Gambar 11 Frame laporan asuransi

Pada Gambar 12 pengguna memilih obat dari tabel lalu tekan tombol pilih. Setelah itu, masukkan jumlah dari obat yang dibelinya kemudian menekan tombol jumlah untuk melihat total yang harus dibayarkan. Lalu masukkan uang yang dibayarkan oleh pasien kemudian menekan tombol OK untuk mendapatkan uang kembalian dan proses pengurangan jumlah obat database akan berlangsung.

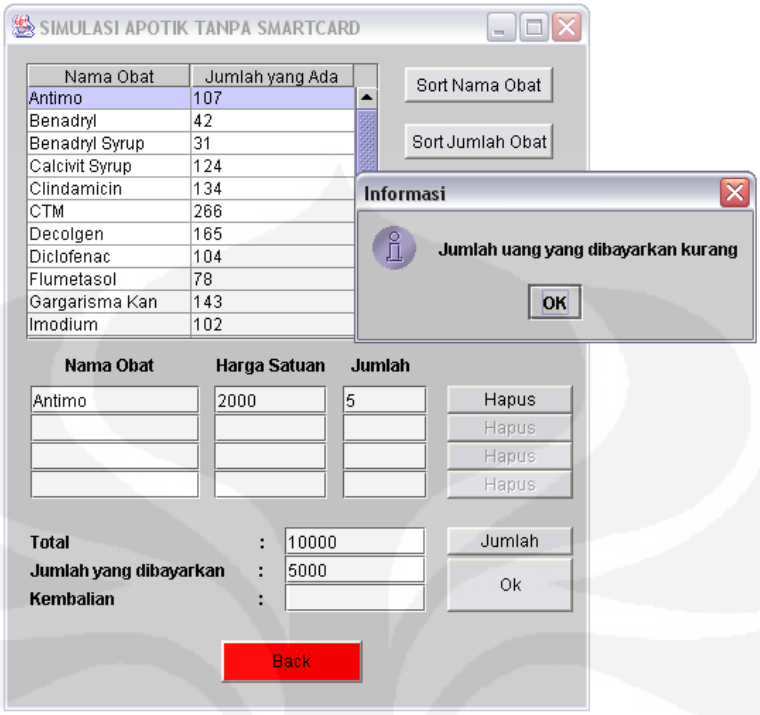


Gambar 12 Frame aplikasi tanpa *smart card*

Pada aplikasi ini terdapat tampilan tampilan jika terjadi kesalahan. Untuk lebih jelasnya pada Gambar 13, Gambar 14 ,dan Gambar 15.

Gambar 13 Kesalahan pada login

Gambar 14 Kesalahan jumlah obat pada aplikasi tanpa *smart card*



Gambar 15 Kesalahan jumlah uang yang dibayarkan pada aplikasi tanpa *smart card*

Pada aplikasi yang menggunakan smart card, apabila kartu belum dimasukkan maka akan keluar tampilan seperti Gambar 16.

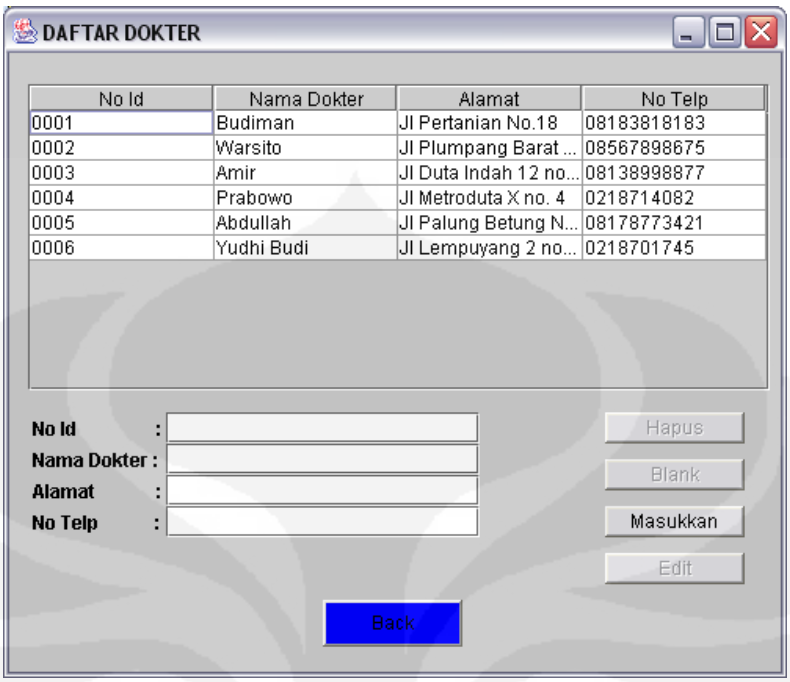
Gambar 16 Pesan kartu belum dimasukkan

Jika kita login sebagai administrator dan telah memasukkan nama dan pin dengan benar maka akan terlihat tampilan seperti ini. Aplikasi ini digunakan untuk menambah atau mengedit data pada database.

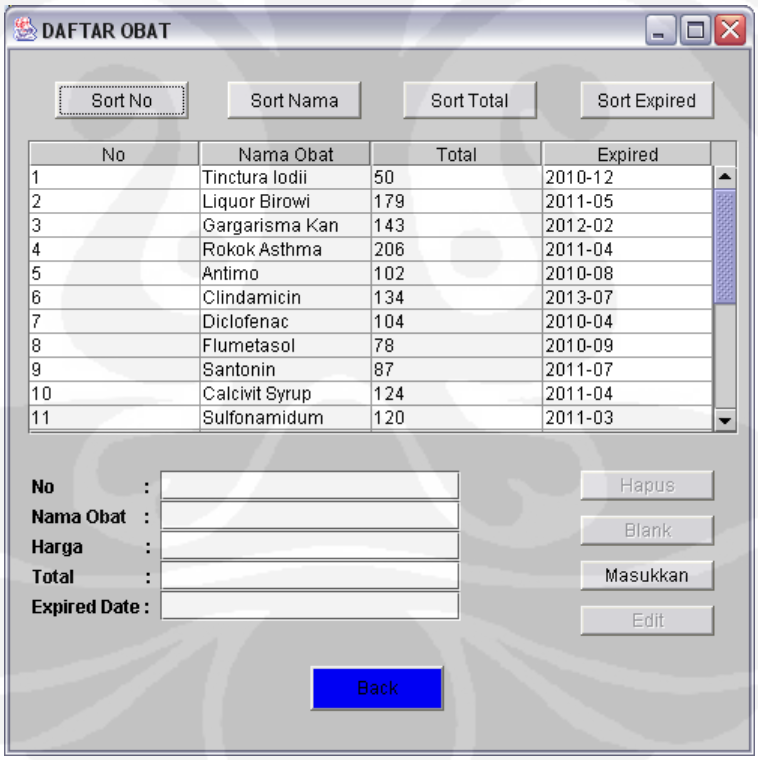
Gambar 17 Tampilan aplikasi administrator

Gambar 18 Tampilan aplikasi edit administrator

Gambar 19 Tampilan aplikasi edit user



Gambar 20 Tampilan aplikasi edit identitas dokter



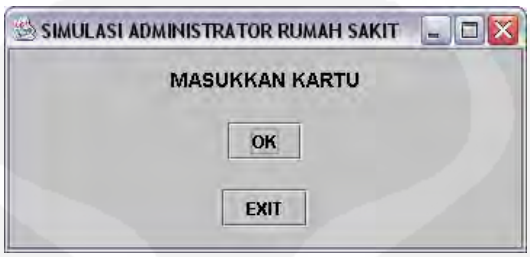
Gambar 21 Tampilan aplikasi edit obat

Apabila pemakaian program simulasi aplikasi apotek telah selesai maka akan muncul tampilan Gambar 22.

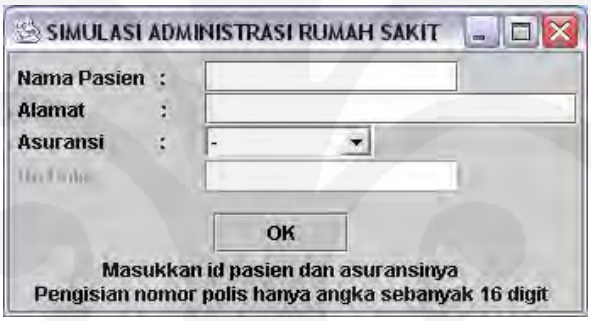
Gambar 22 Tampilan keluar dari program

Lampiran 2 Tampilan Aplikasi Administrasi Rumah Sakit

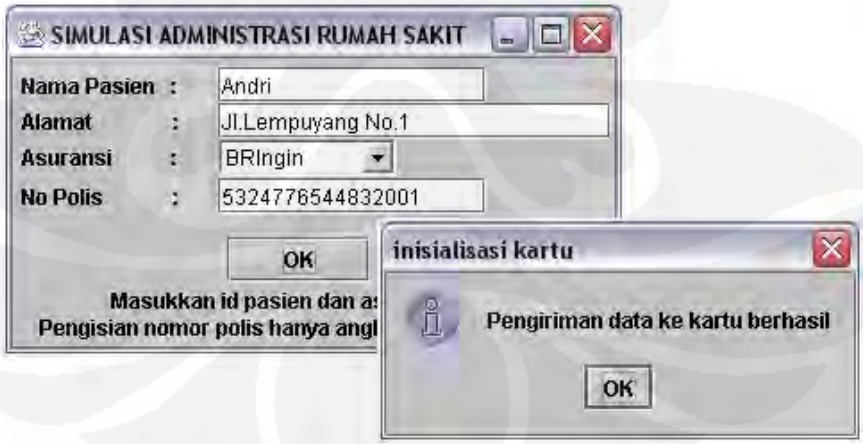
Aplikasi administrasi rumah sakit merupakan aplikasi yang digunakan untuk menginisialisasi kartu pasien yang akan digunakan.



Gambar 23 Tampilan start aplikasi administrasi rumah sakit



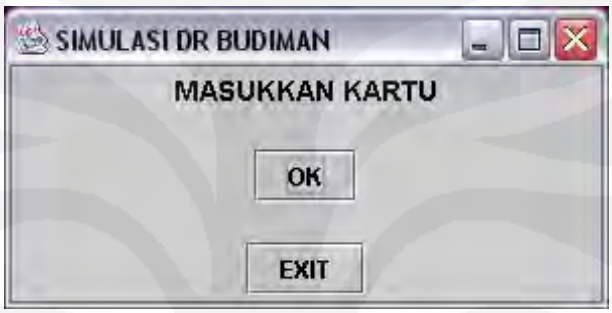
Gambar 24 Tampilan aplikasi administrasi rumah sakit



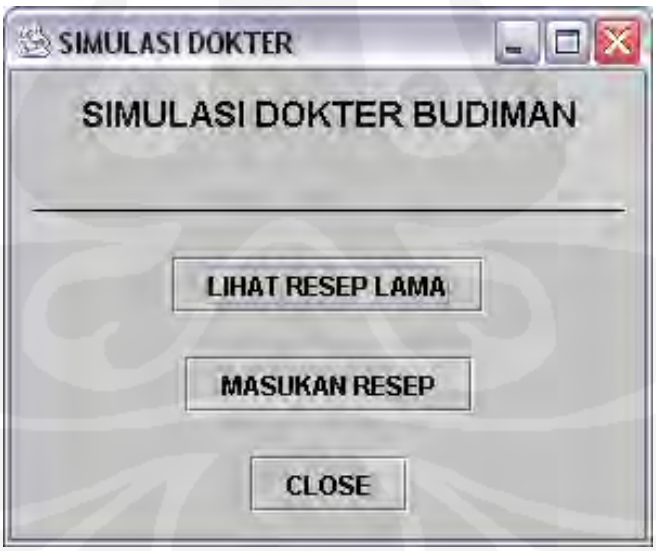
Gambar 25 Tampilan pengiriman data pada aplikasi administrasi rumah sakit

Lampiran 3 Tampilan Aplikasi Dokter

Aplikasi dokter merupakan aplikasi yang digunakan untuk mengisikan obat pada smart card sebagai pengganti kertas resep. Dibawah ini merupakan tampilan pada aplikasi dokter.

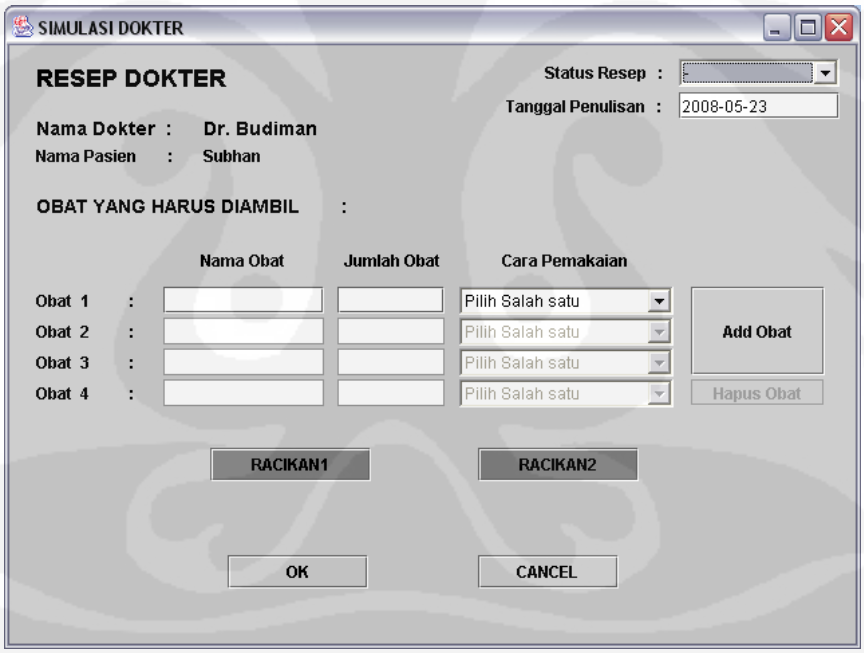


Gambar 26 Tampilan start aplikasi dokter



Gambar 27 Tampilan pilihan aplikasi dokter

Gambar 28 Tampilan resep lama



Gambar 29 Tampilan aplikasi dokter

Dokter dapat mengisi obat racikan dengan menekan tombol racikan 1 atau 2 maka akan keluar tampilan seperti Gambar 30 untuk mengisinya.

Gambar 30 Tampilan obat racikan