RANCANG BANGUN APLIKASI EMAIL CLIENT MENGGUNAKAN JAVA 2 PLATFORM MICRO EDITION (J2ME) PADA MOBILE PHONE

SKRIPSI

Disusun oleh :

DWI ARIANTO

NIM : 06.12.618

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL MALANG

2012

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI EMAIL CLIENT MENGGUNAKAN JAVA 2 PLATFORM MICRO EDITION (J2ME) PADA MOBILE PHONE

SKRIPSI

Disusun dan diajukan untuk melengkapi dan memenuhi persyaratan guna mencapai gelar Sarjana Teknik

Disusun oleh :
DWI ARIANTO
NIM : 06.12.618

Mengetahui,
Ketua Program Studi Teknik Elektro S-1

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y. 1018800189

Diperiksa dan Disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
(Joseph Dedy Irawan, ST, MT)	**(Bima Aulia Firmandani, ST)**
NIP.Y.1974041620011002	**1121**

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK KOMPUTER DAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2012

SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini :

Nama : Dwi Arianto

Nim : 0612618

Program Studi : Teknik Elektro S1

Konsentrasi : Komputer dan Informatika

Dengan ini menyatakan bahwa skripsi yang saya buat adalah hasil karya sendiri, tidak merupakan plagiat dari karya orang lain. Dalam skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sesuai dengan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsi.

Malang, 29 januari 2013

Yang membuat pernyataan

Dwi Arianto

0612618

ABSTRAK

RANCANG BANGUN APLIKASI EMAIL CLIENT MENGGUNAKAN JAVA 2 PLATFORM MICRO EDITION (J2ME) PADA MOBILE PHONE

Dwi Arianto, NIM 06.12.618

Dosen Pembimbing Joseph Dedy Irawan, ST, MT dan Bima Aulia Firmandani, ST

Kebutuhan akan informasi secara cepat, mudah, akurat dan murah sangat dibutuhkan untuk menunjang kegiatan sehari-hari maupun kegiatan perkuliahan. Pemanfaatan email untuk mengirim serta menerima informasi secara cepat masih sangat sedikit, padahal penggunaan email ini lebih murah jika dibandingkan dengan SMS apalagi banyak kampus yang sudah memiliki jaringan internet, sehingga dengan menggunakan email ini biaya yang dikeluarkan sudah menjadi satu dengan biaya internet.

Dalam menyelesaikan permasalahan yang dihadapi, dilakukan penelitian dengan menggunakan metode pengumpulan data dan metode implementasi. Data merupakan sumber atau bahan mentah yang sangat berharga bagi proses menghasilkan informasi. Oleh sebab itu dalam pengambilan data perlu dilakukan penanganan secara cermat dan hati-hati, sehingga data yang diperoleh dapat bermanfaat dan berkualitas.

Dalam merancang aplikasi pada projek ini terlebih dahulu dilakukan pembuatan desain sistem, desain proses, serta desain antar muka aplikasi. Desain proses berguna untuk mengintegrasikan semua proses yang terjadi dalam aplikasi yang akan dibuat. Desain data berguna untuk mengetahui apa saja yang dibutuhkan dalam proses yang akan dikerjakan. Sedangkan perancangan antarmuka berfungsi sebagai antar muka interaksi antara pengguna dengan sistem aplikasi yang dibuat, sehingga pengguna dapat mengoperasikan aplikasi yang dibuat.

Pemanfaatan email untuk mengirim serta menerima informasi secara cepat masih sangat sedikit, padahal penggunaan email ini lebih murah jika dibandingkan dengan SMS apalagi banyak kampus yang sudah memiliki jaringan internet, sehingga dengan menggunakan metode pesan elektrik ini biaya yang dikeluarkan relatif lebih efisien dan murah karena tersedianya fasilitas intranet ditempat-tempat umum.

Aplikasi ini dibuat untuk mempermudah dalam pengiriman pesan melalui pesan elektrik dengan mempergunakan handphone secara cepat dan efisien.

Kata Kunci: aplikasi, email, java, nokia, symbian

BAB VIII

KESIMPULAN, IMPLIKASI, DAN SARAN

Pengantar

Kesimpulan akan menjawab tujuan penelitian yang telah dirumuskan pada bab pendahuluan. Implikasi menguraikan tentang kegunaan penelitian, baik secara teoritis maupun praktis. Saran ditujukan kepada pihak-pihak yang berkepentingan dengan hasil penelitian.

Kesimpulan

Kesimpulan harus menjawab tujuan penelitian yang telah dirumuskan. Jika penelitian ini terdiri atas beberapa tujuan, kesimpulan harus dibuat untuk masing-masing tujuan tersebut.

Implikasi

Implikasi penelitian dapat berupa implikasi teoritis dan implikasi praktis. Implikasi teoritis menyangkut pengembangan ilmu yang diteliti. Implikasi praktis menyangkut kegunaan hasil penelitian untuk pihak-pihak yang berkepentingan dengan hasil penelitian.

Saran

Saran merupakan pernyataan yang berisi anjuran atau usulan mengenai apa yang harus dilakukan sehubungan dengan implikasi penelitian. Saran ditujukan kepada pihak-pihak yang berkepentingan dengan hasil penelitian.

KATA PENGANTAR

Dengan mengucap syukur kepada Tuhan Yang Maha Esa yang dengan segala rahmat dan anugerah-Nya, telah memberikan kekuatan, kesabaran, bimbingan dan perlindungan sehingga penulis dapat menyelesaikan laporan skripsi dengan judul:

RANCANG BANGUN APLIKASI EMAIL CLIENT MENGGUNAKAN JAVA 2 PLATFORM MICRO EDITION (J2ME) PADA MOBILE PHONE

Pembuatan skripsi ini disusun untuk memenuhi syarat akhir kelulusan pendidikan jenjang Strata I di Institut Teknologi Nasional Malang. Dalam penyusunan skripsi ini penulis banyak mendapat bantuan baik moril maupun materil, saran dan dorongan serta semangat dari berbagai pihak, untuk itu penulis mengucapkan terima kasih kepada:

1. Bapak Ir.Yusuf Ismail Nakhoda, MT selaku Ketua Jurusan Teknik Elektro S-1 ITN Malang.
2. Bapak Joseph Dedy Irawan, ST, MT selaku Dosen Pembimbing 1.
3. Bapak Bima Aulia Firmandani, ST selaku Dosen Pembimbing 2.
4. Kedua orangtua, kakak yang telah memberikan dukungan yang tiada hentinya.
5. Untuk adik yang tersayang yang selalu memberikan semangat dan motivasi.
6. Semua teman-teman dan semua pihak yang telah membantu penulis dalam menyelesaikan skripsi ini, tidak bisa penulis sebutkan satu persatu.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat diharapkan.

Akhir kata, penulis mohon maaf kepada semua pihak bilamana selama penyusunan skripsi ini penyusun membuat kesalahan secara sengaja maupun tidak sengaja atau menyinggung pihak lain. Semoga skripsi ini dapat bermanfaat bagi kita semua.

Malang, Agustus 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR


INSTITUT TEKNOLOGI NASIONAL
MALANG
ITN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kebutuhan akan informasi secara cepat, mudah, akurat dan murah sangat dibutuhkan untuk menunjang kegiatan sehari-hari maupun kegiatan perkuliahan. Pemanfaatan email untuk mengirim serta menerima informasi secara cepat masih sangat sedikit, padahal penggunaan email ini lebih murah jika dibandingkan dengan SMS apalagi banyak kampus yang sudah memiliki jaringan internet, sehingga dengan menggunakan email ini biaya yang dikeluarkan sudah menjadi satu dengan biaya internet.

Informasi yang dikirim melalui email ini harus diterima secara cepat seperti halnya SMS agar informasinya benar-benar aktual, untuk itu penggunaan ponsel lebih sesuai daripada menggunakan laptop untuk menerima informasi ini. Untuk saat ini ada satu jenis ponsel yang sudah memiliki fitur untuk menerima email secara cepat, yaitu Blackberry. Fitur push email yang dimiliki Blackberry menjadikan ponsel ini cukup populer karena fitur ini memungkinkan menerima email secara cepat. Harga ponsel Blackberry ini masih cukup mahal sehingga pengguna ponsel ini masih terbatas. Untuk mengatasi permasalahan ini maka diperlukan aplikasi yang dapat digunakan pada ponsel secara umum.

Aplikasi untuk mengambil email dari mail server lebih dikenal sebagai Aplikasi email client. Aplikasi ini sebenarnya sudah banyak digunakan pada computer desktop, seperti Outlook Express, Pegasus Mail, Windows Live Mail, Eudora, Alpine, Blitz Mail dll. Kekurangan dari aplikasi tersebut adalah tertanam pada komputer desktop sehingga kurang mendukung untuk seseorang yang memiliki mobilitas tinggi. Dengan aplikasi

email client pada ponsel untuk memproses informasi perkuliahan diharapkan mahasiswa yang menggunakannya mendapatkan pemberitahuan adanya informasi perkuliahan yang baru secara cepat seperti push email yang dimiliki blackberry.

Aplikasi email client pada ponsel ini dikembangkan menggunakan bahasa pemrograman java. Penggunaan bahasa java karena ponsel yang ada sekarang ini kebanyakan sudah mendukung bahasa pemrograman java ini. Selain itu kelebihan-kelebihan yang dimilikinya seperti portable, robust, berorientasi objek, multithreading, dinamis, terdistribusi, aman, netral secara arsitektur dan masih banyak lagi kelebihan yang dimiliki oleh bahasa pemrograman java ini.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas maka timbul suatu permasalahan bagaimana membuat suatu aplikasi yang dapat menerima dan mengirim pesan melalui pesan elektrik dengan menggunakan jaringan internet sehingga biaya yang dikeluarkan relatif lebih murah karena bisa menggunakan jaringan *intran*et.

1.3 Tujuan

Adapun tujuan yang ingin dicapai dalam pembuatan laporan skripsi ini adalah sebagai berikut:

1. Membuat suatu aplikasi yang dapat memberikan kemudahan dalam mengirim serta menerima pesan elektrik.
2. Untuk mempermudah dalam mengirim dan menerima informasi dengan mempergunakan handphone.

1.4 Batasan Masalah

Batasan masalah yang diambil pada penulisan skripsi ini diharapkan mampu membatasi pembahasan agar sesuai dengan tujuan penelitian itu sendiri. Adapun batasan masalah yang diajukan adalah sebagai berikut :

1. Tugas Akhir ini dirancang dengan menggunakan metode analisi dan perancangan berorientasi objek (*Object-Oriented Analysis and Design*) dan menggunakan bahasa pemodelan UML.
2. Penggunaan aplikasi ini pada ponsel yang sudah mendukung pemrograman Java dengan minimal versi CLDC-1.1 dan MIDP-2.0.
3. Tidak membahas *setting* koneksi ponsel ke internet karena untuk *setting* tersebut sangat tergantung dari jenis ponsel dan operator seluler yang digunakan.
4. Aplikasi akan dikembangkan menggunakan JME (*Java Micro Edition*) SDK 3.0 dan LWUIT (*Lightweight User Interface Toolkit*) untuk tampilan grafisnya.

1.5 Metode Penelitian

Dalam menyelesaikan permasalahan yang dihadapi, dilakukan penelitian dengan menggunakan metode pengumpulan data dan metode implementasi.

1.5.1 Metode Pengumpulan Data

Data merupakan sumber atau bahan mentah yang sangat berharga bagi proses menghasilkan informasi. Oleh sebab itu dalam pengambilan data perlu dilakukan penanganan secara cermat dan hati-hati, sehingga data yang diperoleh dapat bermanfaat dan berkualitas.

Dalam pengumpulan data penyusun mengunakan metode sebagai berikut :

1. Studi Lapangan

   Dengan metode ini data-data diperoleh langsung dari sumber yang bersangkutan, dimana peneliti berhadapan langsung dengan obyek yang diteliti, yang dilakukan dengan cara :

   a. Survey

Teknik pengumpulan data dengan cara terjun secara langsung dan mencatat secara sistematis terhadap obyek masalah.

b. Wawancara / Interview

Teknik pengumpulan data dengan jalan mengadakan komunikasi langsung dengan pihak yang bersangkutan.

2. Studi Pustaka / Literatur

Pengumpulan data ini dilakukan dengan cara mencari bahan-bahan kepustakaan sebagai landasan teori yang ada hubunganya dengan permasalahan yang dijadikan obyek penelitian.

1.5.2 Metode Implementasi

Dalam implementasi sistem untuk menerima dan mengirim pesan melalui pesan elektrik melalui hanphone yang harus dilakukan antara lain :

1. Studi Literatur.

   Pengumpulan data yang dilakukan dengan mencari bahan-bahan kepustakaan dan referensi dari berbagai sumber sebagai landasan teori yang ada hubunganya dengan permasalahan yang dijadikan objek penelitian.

2. Analisa Kebutuhan Sistem.

   Data dan informasi yang telah diperoleh akan dianalisa agar didapatkan kerangka global yang bertujuan untuk mendefinisikan kebutuhan sistem baik hardware maupun software, di mana nantinya akan digunakan sebagai acuan perancangan sistem.

3. Perancangan Sistem.

   Perancangan sistem yang dilakukan dengan membuat alur sistem untuk mengetahui alur jalannya aplikasi yang dibuat.

4. Desain Aplikasi.

   Untuk menghasilkan desain aplikasi yang baik, harus dibuat secara sederhana tanpa meninggalkan aspek tampilan yang menarik

5. Pengujian Aplikasi.

   Pengujian dilakukan untuk mengetahui hasil dari sistem yang telah dibuat seperti pengujian aplikasi, dan delay.

Tabel 1 Pengumpulan data dengan cara tanya secara langsung dan mencatat secara sistematis tentang objek penelitian.

b. Wawancara / Interview

Pengumpulan data dengan tatap muka langsung komunikasi langsung dengan pihak yang berwenang.

2. Studi Pustaka / Literatur

Pengumpulan data ini dilakukan dengan cara mencari bahan-bahan kepustakaan sebagai landasan teori yang erat kaitannya dengan permasalahan yang dibahas sebagai pendukung.

1.5.3 Metode Implementasi

Dalam implementasi sistem untuk memudahkan dan mengidentifikasi pada pasar elektronik menjalankan beberapa yang harus dilakukan antara lain :

1. Studi Literatur

Mencari dan mempelajari dengan mencari bahan-bahan kepustakaan dan referensi dari berbagai sumber sebagai landasan teori yang ada hubungannya dengan permasalahan yang dibahas.

2. Analisa Kebutuhan Sistem

Hasil dari informasi yang telah dipercaya akan dianalisa agar didapatkan kerangka global yang berupaya untuk memudahkan kebutuhan sistem baik kebutuhan masukan keluaran. Di mana nantinya akan digunakan sebagai acuan pembangun sistem.

3. Perancangan Sistem

Perancangan sistem yang dibutuhkan dengan membuat alur sistem untuk mengetahui alur informasi yang akan

4. Desain Aplikasi

Untuk mengimplikasikan desain aplikasi yang baik, harus dibuat secara sederhana tetapi menghasilkan aspek kemudahan yang maksimal.

5. Pengujian Aplikasi

Pengujian dilakukan untuk mengetahui hasil dari sistem yang telah dibuat sesuai perancangan atau desain.

6. Pemeliharaan (*maintenance*)

   Oleh karena kebutuhan pemakai selalu meningkat, maka perangkat lunak yang telah selesai perlu dip elih ara agar dapat mengantisipasi kebutuhan pemakai terhadap fungsi-fungsi baru yang berasal dari luar atau perubahan-perubahan pada sistem yang dapat timbul karena munculnya sistem operasi baru, perangkat keras baru, dan sebagainya.

1.6 Sistematika Penulisan

Sistematika penulisan yang diuraikan dalam penyusunan skripsi ini adalah sebagai berikut :

BAB I : PENDAHULUAN

Bab ini berisi tentang latar belakang, tujuan, permasalahan, batasan masalah, dan sistematika pembahasan dari skripsi ini.

BAB II : TINJAUAN PUSTAKA

Bab ini berisi penjelasan tentang teori – teori yang mendukung dalam implementasi system yang meliputi teori Email, E-Mail, MIDP, CLDC.

BAB III : PERANCANGAN SISTEM

Bab ini berisi tentang serangkaian langkah yang ditempuh dalam pelaksanaan kegiatan penelitian ini serta langkah perancangan model sistem yang diusulkan dari awal sampai akhir, rancangan struktur database yang dipergunakan dalam aplikasi serta desain masukan dan keluaran aplikasi (*design user interface*).

BAB IV : HASIL DAN ANALISA

Bab ini berisi pembahasan hasil pengujian dan analisa mengenai cara kerja dari aplikasi.

BAB V : PENUTUP

Bab ini berisi kesimpulan dan saran dari hasil pembahasan pada skripsi ini.

BAB II
LANDASAN TEORI

2.1 Aplikasi

Perangkat lunak aplikasi yaitu perangkat lunak yang digunakan untuk membantu pemakai komputer untuk melaksanakan pekerjaannya. Jika ingin mengembangkan program aplikasi sendiri, maka untuk menulis program aplikasi tersebut, dibutuhkan suatu bahasa pemrograman, yaitu language software, yang dapat berbentuk assembler, compiler ataupun interpreter. Jadi language software merupakan bahasanya dan program yang ditulis merupakan program aplikasinya.

2.2 Internet

Internet (kependekan dari *interconnection-networking*) secara harfiah ialah sistem global dari seluruh jaringan komputer yang saling terhubung menggunakan standar Internet Protocol Suite (TCP/IP) untuk melayani miliaran pengguna di seluruh dunia. Manakala Internet (huruf 'I' besar) ialah sistem komputer umum, yang berhubung secara global dan menggunakan TCP/IP sebagai protokol pertukaran paket (*packet switching communication protocol*). Rangkaian internet yang terbesar dinamakan **Internet**. Cara menghubungkan rangkaian dengan kaedah ini dinamakan *internetworking*. [6]

2.3 Mobile Phone

Telepon seluler (ponsel) atau telepon genggam (telgam) atau *handphone* (HP) atau disebut pula adalah perangkat telekomunikasi elektronik yang mempunyai kemampuan dasar yang sama dengan telepon konvensional saluran tetap, namun dapat dibawa ke mana-mana (portabel, *mobile*) dan tidak perlu disambungkan dengan jaringan telepon menggunakan kabel (nirkabel; *wireless*). Saat ini Indonesia mempunyai dua jaringan telepon nirkabel yaitu sistem GSM (*Global System for Mobile Telecommunications*) dan sistem CDMA (*Code Division Multiple Access*). Badan yang mengatur telekomunikasi seluler Indonesia adalah Asosiasi Telekomunikasi Seluler Indonesia (ATSI).

2.4 E-mail (*electronic mail*)

Email (Electronik mail) adalah media komunikasi yang memungkinkan pertukaran informasi berbasis text untuk dipertukarkan secara elektronik. Seiring

dengan perkembangan teknologi, email kemudian memungkinkan pertukaran file digital melalui lampiran (*attachment*). *Email* merupakan sebuah layanan pengiriman surat elektronik yang dikirim melalui internet. *Email* dikirim dari suatu alamat *email* yang terdapat pada sebuah *mail server* kepada alamat *email* yang lainnya yang terdapat pada *mail server* yang sama maupun pada *mail server* yang berbeda. *Email* dapat dianalogikan dengan kotak surat yang ada di kantor POS sedangkan *server email* dapat diibaratkan sebagai kantor POS. Dengan analogi ini sebuah *mail server* dapat memiliki banyak *account email* yang ada didalamnya.

Perangkat mobile phone elektronik telekomunikasi, sering disebut sebagai telepon seluler atau ponsel. Ponsel terhubung ke jaringan komunikasi nirkabel melalui gelombang radio atau transmisi satelit. Kebanyakan telepon seluler menyediakan komunikasi suara, *Short Message Service* (SMS), *Multimedia Message Service* (MMS), dan juga menyediakan layanan Internet seperti browsing Web dan e-mail.

2.5 SMTP *(Simple Mail Transfer Protocol)*

Simple Mail Transfer Protocol (SMTP)Simple Mail Transfer Protocol (SMTP) adalah suatu protokol yang digunakan untuk mengirimkan pesan e-mail antar server, yang bisa dianalogikan sebagai kantor pos. Ketika kita mengirim sebuah e-mail, komputer kita akan mengarahkan e-mail tersebut ke sebuah SMTP server, untuk diteruskan ke mail-server tujuan

2.6 POP 3 *(Post Office Protocol version 3)*

POP3 adalah kepanjangan dari *Post Office Protocol version 3*, yakni protokol yang digunakan untuk mengambil email dari email server. Protokol POP3 dibuat karena desain dari sistem email yang mengharuskan adanya email server yang menampung email untuk sementara sampai email tersebut diambil oleh penerima yang berhak. Kehadiran email server ini disebabkan kenyataan hanya sebagian kecil dari komputer penerima email yang terus-menerus melakukan koneksi ke jaringan internet.

2.7 IMAP *(Internet Message Access Protocol)*

IMAP (Internet Message Access Protocol) adalah protokol standar untuk mengakses/mengambil e-mail dari server. IMAP memungkinkan pengguna memilih pesan e-mail yang akan ia ambil, membuat folder di server, mencari pesan e-mail tertentu, bahkan menghapus pesan e-mail yang ada.

2.8 Perbedaan IMAP dan POP3

POP3 protokol mengasumsikan bahwa hanya ada satu klien tersambung ke kotak surat. Sebaliknya, protokol IMAP memungkinkan akses bersamaan oleh banyak klien. IMAP sangat cocok untuk Anda jika kotak surat Anda adalah tentang untuk dikelola oleh beberapa pengguna.

2.9 SSL *(Secure Socket Layer)*

Secure Sockets Layer atau yang disingkat SSL adalah sebuah protokol keamanan data yang digunakan untuk menjaga pengiriman data antara web server dan pengguna situs web tersebut. SSL umumnya sudah terinstall didalam mayoritas browser web yang ada (IE, Netscape, Firefox, dll), sehingga pengguna situs web dapat mengidentifikasi tingkat keamanan situs web tersebut yang menggunakan protokol keamanan SSL ini.

Browser web secara otomatis akan mencek apakah sertifikat SSL dan identitas situs web valid dan situs tersebut terdaftar pada otoritas sertifikasi (CA) SSL (cth. Verisign). Dengan demikian, SSL ini menjadi sangat penting terutama untuk situs web yang menjalankan transaksi online.

Koneksi SSL akan memproteksi informasi vital dengan meng-enkripsi informasi yang dikirim dan diterima antara pc pengguna situs dan web server, sehingga informasi yang berjalan tidak mungkin dapat diambil ditengah jalan dan dibaca isinya. Hal ini berarti pengguna tidak perlu ragu untuk mengirim informasi vital seperti nomor kartu kredit kepada situs web yang telah memasang SSL tersertifikat ini.

2.9.1. Cara Kerja *Secures Socket Layer* (SSL)

Seorang pelanggan masuk kedalam situs anda dan melakukan akses ke URL yang terproteksi (ditandai dengan awalan https atau dengan munculnya pesan dari browser).

Server anda akan memberitahukan secara otomatis kepada pelanggan tersebut mengenai sertifikat digital situs anda yang menyatakan bahwa situs anda telah tervalidasi sebagai situs yang menggunakan SSL.

Browser pelanggan akan mengacak "session key" dengan "public key" situs anda sehingga hanya situs anda yang akan dapat membaca semua transaksi yang terjadi antara browser pelanggan dengan situs anda.

Hal diatas semua terjadi dalam hitungan detik dan tidak memerlukan aktifitas apapun dari pelanggan.

2.9.2. Keuntungan Menggunakan SSL

Transaksi Bisnis ke Bisnis atau Bisnis ke Pelanggan yang tidak terbatas dan menambah tingkat kepercayaan pelanggan untuk melakukan transaksi online dari situs anda.

2.9.3. Layanan SSL

Berikut adalah jenis layanan SSL yang kami tawarkan kepada para pemilik situs transaksi online.

1. Quick Validatio SSl Certificates

   Quick validation memproses kepemilikan sertifikat SSL dengan verifikasi melalui email kepada pemilik yang terdaftar pada informasi kontak domain WHOIS. Sertifikat ini juga memastikan bahwa data yang dikirim menggunakan teknologi keamanan enkripsi.
2. SBS Instant : Rp. 299.500 / tahun

   Enkripsi 128bit, prosedur validasi otomatis, digunakan hanya untuk 1 domain, tanpa garansi, tanpa SBS trust logo. Paling cocok untuk transaksi online skala kecil atau perusahan yang membutuhkan SSL secara cepat dan mudah.
3. GeoTrust Quick SSL : Rp. 1.250.000 / tahun

   Enkripsi 128bit, 2 faktor prosedur validasi, digunakan hanya untuk 1 domain, garansi GeoTrust $10,000. Paling cocok untuk transaksi online skala kecil atau perusahan yang membutuhkan SSL secara cepat dan mudah.
4. GeoTrust Quick SSL Premium : Rp. 1.750.000 / tahun

   Enkripsi 128bit, 2 faktor prosedur validasi, digunakan hanya untuk 1 domain, support Mobile Device dan Smartphones, garansi GeoTrust $100,000, SSL seal menampilkan jam dan tanggal. Cocok untuk transaksi online skala menengah dgn tingkat keamanan tinggi.
5. Rapid SSL : Rp. 290.000 / tahun

Enkripsi 128bit, prosedur validasi otomatis, digunakan hanya untuk 1 domain, garansi $10,000. Cocok untuk transaksi online skala kecil atau perusahan yang membutuhkan SSL secara cepat dan mudah.

6. Full Validation SSL Certificates

   Full validation memproses kepemilikan sertifikat SSL dengan verifikasi melalui email kepada pemilik yang terdaftar pada informasi kontak domain WHOIS dan juga melakukan verifikasi secara manual dengan langsung membuat kontak dengan pemilik sertifikat atas identitas bisnisnya. Sertifikat ini juga memastikan bahwa data yang dikirim menggunakan teknologi keamanan enkripsi.

7. SBS Secure : Rp. 790.000 / tahun

   Enkripsi 128bit, prosedur validasi manual terhadap kepemilikan domain dan identitas bisnis, digunakan hanya untuk 1 domain, garansi SBS $75,000, tanpa SBS trust logo.

8. SBS Secure Plus : Rp. 1.190.000 / tahun

   Enkripsi 128bit, prosedur validasi manual terhadap kepemilikan domain dan identitas bisnis, digunakan hanya untuk 1 domain, garansi SBS $1,000,000, dapat memasang SBS trust logo.

9. GeoTrust SSL True BusinessID : Rp. 2.400.000 / tahun

   Enkripsi 128bit, manual verifikasi terhadap identitas bisnis dan verifikasi kepemilikan domain, digunakan hanya untuk 1 domain, garansi GeoTrust $10,000, SSL seal menampilkan jam, tanggal dan nama perusahaan anda.

2.10 Port

Dalam protokol jaringan TCP/IP, sebuah ***port*** adalah mekanisme yang mengizinkan sebuah komputer untuk mendukung beberapa sesi koneksi dengan komputer lainnya dan program di dalam jaringan. Port dapat mengidentifikasikan aplikasi dan layanan yang menggunakan koneksi di dalam jaringan TCP/IP. Sehingga, port juga mengidentifikasikan sebuah proses tertentu di mana sebuah server dapat memberikan sebuah layanan kepada klien atau bagaimana sebuah klien dapat mengakses sebuah layanan yang ada dalam server.

2.11 Java

Java adalah sebuah teknologi yang diperkenalkan oleh Sun Microsystems pada pertengahan tahun 1990. Menurut definisi dari Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. Kita lebih menyukai menyebut Java sebagai sebuah teknologi dibanding hanya sebuah bahasa pemrograman, karena Java lebih lengkap dibanding sebuah bahasa pemrograman konvensional. Teknologi Java memiliki tiga komponen penting, yaitu: [5]

1. *Programming-language specification*
2. *Application-programming interface*
3. *Virtual-machine specification*

2.12 Java API

Komponen ini merupakan kumpulan *library* untuk menjalankan dan mengembangkan program Java pada perangkat bergerak. Java API terdiri dari tiga bagian utama:

1. Java Standard Edition (SE), sebuah standar API untuk merancang aplikasi desktop dan applets dengan bahasa dasar yang mendukung grafis, M/K, keamanan, konektivitas basis data dan jaringan.
2. Java Enterprise Edition (EE), sebuah inisiatif API untuk merancang aplikasi server dengan mendukung untuk basis data.
3. Java Macro Edition (ME), sebuah API untuk merancang aplikasi yang jalan pada alat kecil seperti telepon genggam, komputer genggam dan pager.

Dalam API itu terdapat fungsi-fungsi/perintah-perintah untuk menggantikan bahasa yang digunakan dalam *system calls* dengan bahasa yang lebih terstruktur dan mudah dimengerti oleh *programmer*. Fungsi yang dibuat dengan menggunakan API tersebut kemudian akan memanggil *system calls* sesuai dengan sistem operasinya. Tidak tertutup kemungkinan nama dari *system calls* sama dengan nama di API. Keuntungan memprogram dengan menggunakan API adalah:

1. Portabilitas.

   Programmer yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah ter- *install* API tersebut. Sedangkan

system call berbeda antar sistem operasi, dengan catatan dalam implementasinya mungkin saja berbeda.

2. Lebih Mudah Dimengerti.
   API menggunakan bahasa yang lebih terstruktur dan mudah dimengerti daripada bahasa *system call*. Hal ini sangat penting dalam hal editing dan pengembangan.

System call interface ini berfungsi sebagai penghubung antara API dan *system call* yang dimengerti oleh sistem operasi. *System call interface* ini akan menerjemahkan perintah dalam API dan kemudian akan memanggil *system calls* yang diperlukan. Untuk membuka suatu *file* tersebut *user* menggunakan program yang telah dibuat dengan menggunakan bantuan API, maka perintah dari *user* tersebut diterjemahkan dulu oleh program menjadi perintah *open()*. Perintah *open()* ini merupakan perintah dari API dan bukan perintah yang langsung dimengerti oleh kernel sistem operasi. Oleh karena itu, agar keinginan *user* dapat dimengerti oleh sistem operasi, maka perintah *open()* tadi diterjemahkan ke dalam bentuk *system call* oleh *system call interface*. Implementasi perintah *open()* tadi bisa bermacam-macam tergantung dari sistem operasi yang kita gunakan.

Cara Memakai API :

1. Dilakukan dengan mengimpor package/kelas import java.util.Stack;
2. Ada beberapa kelas bernama sama dipackage yang berbeda, yaitu :
   a. import salah satu dan gunakan nama lengkap untuk yang lain, atau
   b. gunakan nama lengkap semua kelas

Ada tiga jenis Bahasa Pemrograman Java Application Programming Interface (API), inti resmi Java API, yang terdapat dalam JDK atau JRE, dari salah satu edisi dari Java Platform. Tiga edisi dari Java Platform adalah Java ME (Micro edition), Java SE (Standard edition), dan Java EE (Enterprise edition).

a. Resmi opsional API yang dapat didownload secara terpisah. Spesifikasi API ini didefinisikan sesuai dengan Spesifikasi Jawa Request (JSR), dan kadang-kadang beberapa API ini kemudian dimasukkan dalam API inti dari platform (contoh yang paling terkenal dari jenis ini adalah swing).
b. API tidak resmi, yang dikembangkan oleh pihak ketiga, tetapi tidak berkaitan dengan JSRs apapun.

Pihak-pihak ketiga dapat dengan bebas mengimplementasikan spesifikasi JSR API resmi (bahkan untuk API inti dari bahasa), memberikan bahwa mereka menyesuaikan diri dengan Teknologi Kompatibilitas Kit (TCK) untuk JSR ini (yang TCK adalah paket tes yang memeriksa kesesuaian dari implementasi untuk JSR). The result of this freedom is that many official APIs have more implementations than the Sun's Reference implementation (RI). Hasil dari kebebasan ini adalah bahwa banyak API resmi memiliki lebih implementasi daripada Matahari pelaksanaan Referensi (RI). Anda dapat menggunakan kelas Java dan API untuk mengakses konten di berbagai server konten.

The DB2 Content Management pusat informasi yang menyediakan dokumentasi API berikut:

1. Kelas Java dan API
2. JavaBeans
3. C + + kelas dan API
4. Tabel kontrol system
5. Perpustakaan pengguna server keluar
6. DB2 Content Manager aturan sistem konektor

Berikut ini adalah sebagian daftar Application Programming Interface (API) untuk Java Programming Language.

1. Java Platform, Standard Edition (Java SE)
2. Bundled 1.1.1 API (bagian dari standar download)
3. Opsional 1.1.2 API (download terpisah)* Java Platform, Enterprise Edition (Java EE)
4. Bundled 1.2.1 API (bagian dari standar download)
5. Opsional 1.2.2 API (download terpisah)
6. Java Platform, Micro Edition (Java ME)
7. Unofficial API (Dirilis oleh pihak ketiga)

2.12.1. JavaSE (Java Standard Edition)

Java Platform, Standard Edition memungkinkan Anda mengembangkan dan menyebarkan aplikasi Java pada desktop dan server, serta hari ini menuntut Embedded dan lingkungan Real-Time. Java SE meliputi kelas yang mendukung pengembangan

Java Web Services dan memberikan dasar untuk Java Platform, Enterprise Edition (Java EE).

2.12.2. JavaEE (Java Enterprise Edition)

Java EE adalah platform terdepan untuk pengembangan dan penggunaan aplikasi web dan enterprise. Java EE memperkenalkan fitur-fitur untuk meningkatkan fleksibilitas platform dan memudahkan perusahaan untuk menggunakan skenario aplikasi khusus, selain platform enterprise lengkap, untuk membantu memenuhi kebutuhan penggunanya. Platform Java EE dan spesifikasi teknologi intinya terus dikembangkan melalui Java Community Process (SM) (JCP) dan selain itu, melalui beragam komunitas open source. JCP adalah sebuah upaya komunitas bersama, yang mencakup perusahaan dan organisasi besar terkemuka di industrinya, (termasuk Apache, Caucho, Eclipse, Fujitsu, Google, HP, IBM, Oracle, Red Hat dan SAP AG) bersama anggota komunitas independen.

2.12.3. JavaME (Java Micro Edition)

JavaME (*Java Micro Edition*) atau biasa disebut JME adalah lingkungan pengembangan yang didesain untuk meletakkan perangkat lunak java pada perangkat elektronik. Pada JME, jika perangkat lunak berfungsi baik pada sebuah perangkat maka belum tentu berfungsi baik pada perangkat yang lainya. JME biasanya digunakan pada telepon seluler, pager, personal digital assistants (PDA's) dan sejenisnya. JME adalah bagian dari JSE, karena itu tidak semua *library* yang ada pada JSE dapat digunakan pada JME. Tetapi JME mempunyai beberapa *library* khusus yang tidak dimiliki JSE. Arsitektur JME dapat dilihat pada gambar 2.1.

Seperti yang telah disebutkan sebelumnya, J2ME dirancang untuk dapat menjalankan program Java pada perangkat-perangkat semacam telepon selular atau PDA yang memiliki karakteristik yang berbeda dengan sebuah computer biasa (PC), misalnya kecilnya jumlah memori pada telepon selular dan PDA jika dibandingkan dengan jumlah memori yang tersedia pada sebuah komputer (PC). J2ME terdiri atas komponen-komponen sebagai berikut (Wicaksono, 2002, p6) :

1. *Java Virtual Machine* (JVM)

   Komponen ini untuk menjalankan program-program Java pada *emulator* atau langsung pada perangkat bergerak.

2. Java API

Komponen ini merupakan kumpulan *library* untuk menjalankan dan mengembangkan program Java pada perangkat bergerak.

3. *Tools* lain untuk pengembangan aplikasi Java, semacam *emulator* JavaPhone, emulator Motorola, emulator Sony Erickson, dan lain-lain.

Tabel 2.1 Java Profile

<table>
<tr><td colspan="2">Profile</td></tr>
<tr><td rowspan="2">Configuration</td><td>Kumpulan Library</td></tr>
<tr><td>JVM</td></tr>
<tr><td colspan="2">Sistem Operasi</td></tr>
</table>

2.13 Mobile Information Device Profile (MIDP)

MIDP (Mobile Information Device Profile), adalah seperangkat API *(Application Programming Interface)* dari Java, yang umumnya digunakan pada CLDC (*Connected Limited Device Configuration).* MIDP (Mobile Information Device Profile) menyediakan dasar J2ME *(Java 2 MicroEdition)* application runtime environment, yang digunakan pada perangkat informasi mobile, seperti ponsel dan pager dua arah. Spesifikasi MIDP (Mobile Information Device Profile) mencakup antara lain interface, penyimpanan yang mantap, networking, dan model aplikasi.

2.14 J2ME

2.14.1 J2ME *Configurations*

J2ME dibagi menjadi dua buah bagian yang dikenal dengan istilah *configuration* dan *profile* (Wicaksono, 2002, p7), yaitu :

Berdasarkan spesifikasi perangkat kerasnya, J2ME memiliki 2 macam konfigurasi yaitu CLDC *(Connected Limited Device Configuration)* dan CDC *(Connected Device Configurations)*. Perbedaan tersebut dapat dilihat pada tabel berikut:

a) *Connected Limited Device Configuration* (CLDC)

Kategori ini umumnya digunakan untuk aplikasi Java pada perangkatperangkat bergerak dengan ukuran memori 160-512 kiloBytes, misalnya telepon selular.

b) *Connected Device Configuration* (CDC)

Kategori ini umumnya digunakan untuk aplikasi Java pada perangkat-perangkat bergerak dengan ukuran memori minimal 2 MegaBytes, misalnya PDA dan *PDA Phone*.

CLDC	CDC
Mengimplementasikan sebagian fitur dari J2SE	Mengimplementasikan seluruh fitur dari J2SE
Menggunakan KVM (Kilobyte Virtual Machine)	Menggunakan CVM (C-Virtual Machine)
Digunakan pada handphone, PDA, pager yang memiliki memori terbatas (160-512 kb)	Digunakan pada Internet TV, Nokia Communicator yang memiliki memori minimal 2MB
Prosesor 16 / 32 bit	Prosesor 32 bit

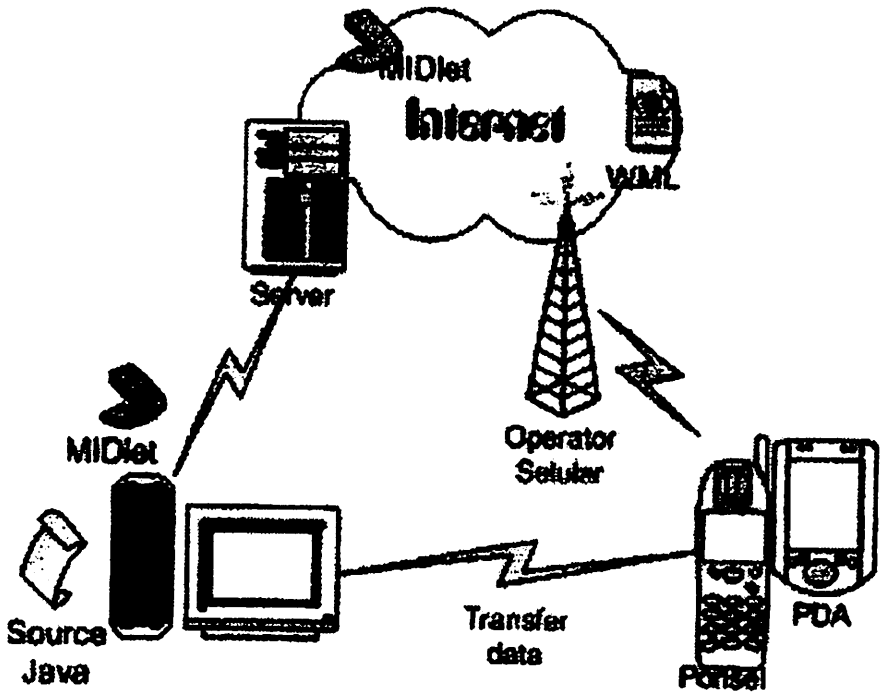
Gambar 2.1 Perbandingan CLDC dan CDC

2.14.2 *J2ME Profile*

J2ME Profile mengimplementasikan kemampuan atau fitur yang sangat tergantung ada jenis dan tipe perangkat bergerak yang digunakan. Ada lima kategori *J2ME Profile* saat ini, yakni *Mobile Information Device Profile* (MIDP), *Foundation Profile* (FP), *Personal Profile*, *RMI Profile*, *PDA Profile*. MIDP menyediakan *libraries* Java untuk implementasi dasar antarmuka (GUI), implementasi jaringan (*networking*), *database*, dan *timer*. MIDP dirancang khusus untuk *wireless phone* dan *pager*.

2.15 MIDlet

MIDlet adalah aplikasi Java yang ditulis untuk peralatan MID. Karena ditulis dengan bahasa Java yang memiliki portabilitas tinggi, maka pe-ngembangan aplikasi ini dapat dilakukan pada PC dan hasil kompilasi-nya dapat di-install ke peralatan MID baik melalui transfer data maupun download melalui internet dengan OTA *(Over The Air) Provisioning*. [2]



Gambar 2.2 Siklus MIDlet

Yang didistribusikan itu sendiri adalah MIDlet yang sudah dikemas dalam paket JAR (Java Archieve) berisi seluruh class MIDlet beserta file resource-nya. File JAR ini didampingi oleh sebuah file JAD yang ber isi deskripsi file JAR yang bersangkutan. Informasi yang terkandung di dalamnya berupa ukuran file, versi, alamat, pengembang dsb.

Distribusi melalui OTA adalah distribusi yang sangat disarankan untuk pembuatan MIDlet bagi keperluan komersial dan pemakaian yang luas. Dalam prakteknya distribusi dengan cara ini digabungkan dengan tekno logi WAP sehingga pengguna diberi kesempatan untuk melakukan browsing ke situs pengembang MIDlet melalui handset-nya.

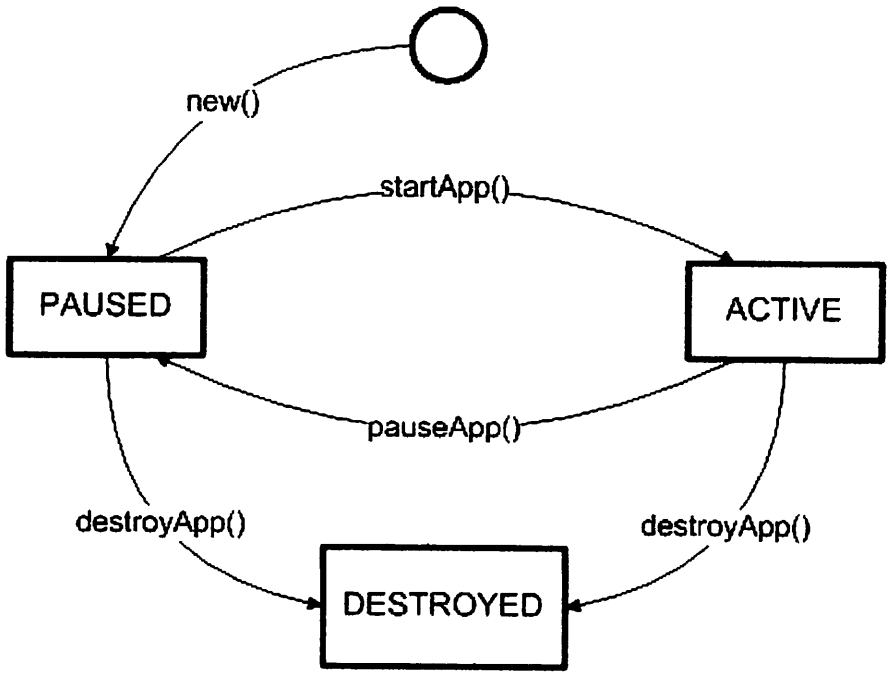
Dalam halaman WAP, pengguna dapat memilih MIDlet apa saja yang akan didownload. Pengembang dapat pula menyisipkan gambar screen shot dari MIDletnya pada halaman WAP tersebut sehingga mempermu-dah pengguna dalam memilih MIDlet.

Ketika pengguna memilih untuk men-download suatu MIDlet maka Application Management System (AMS) yang terdapat pada handset pengguna akan membaca

informasi mengenai MIDlet tersebut yang terdapat pada file JAD. Karena file ini sangat kecil maka proses ini tidak akan membebani pengguna dengan biaya tinggi.

Informasi yang diambil oleh AMS adalah lokasi MIDlet beserta ukuran-nya. AMS akan mengkonfirmasi pengguna untuk men-download MIDlet tersebut atau membatalkannya. Jika pengguna memilih untuk melaku-kan download, maka AMS segera men-downloadfile JAR untuk disim-pan pada memory handset pengguna.

Menurut Feng dan Zhu (2001, p56), MIDlet adalah aplikasi yang dibuat menggunakan J2ME dengan *profile* MIDP. Kemampuan dikhususkan untuk digunakan pada perangkat bergerak dengan kemampuan CPU, memori, *keyboard*, dan layar yang terbatas, misalnya pada telepon selular, pager, PDA, dan sebagainya. Secara umum, terdapat beberapa hal penting dalam membuat sebuah aplikasi MIDlet, yaitu menyangkut *lifecycle*, *user interface*, *command handling*, *deployment*, dan *application management*. *Lifecycle* dari sebuah MIDlet ditangani oleh *Application Management Perangkat lunak* (AMS). AMS ini adalah sebuah lingkungan tempat siklus dari sebuah MIDlet mampu diciptakan, dijalankan, dihentikan, maupun dihapus. AMS sering pula dinamakan dengan *Java Application Manager* (JAM). MIDlet memiliki beberapa *state*, yaitu *Paused*, *Active*, *Destroyed*. Ketika masing-masing *state* dipanggil, beberapa *method* standar yang bersesuaian dipanggil. Untuk lebih jelasnya dapat dilihat pada Gambar 2.19.



Gambar 2.3 Skema MIDlet

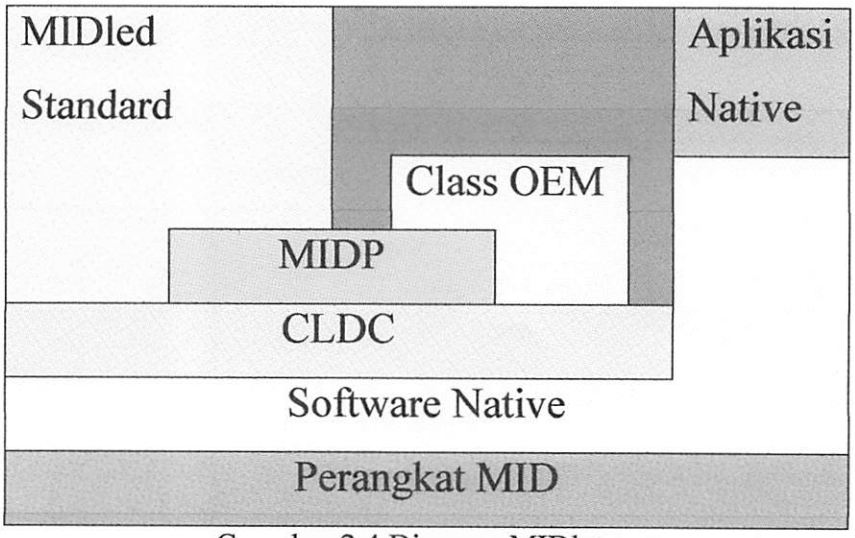
Ketika MIDlet pertama kali diciptakan dan diinisialisasi, MIDlet akan berada dalam *state* "*Paused*". Apabila terjadi kesalahan selama konstruksi MIDlet, MIDlet akan berpindah ke *state* "*Destroyed*", dan MIDlet batal diciptakan dengan jalan memanggil fungsi standar *destroyApp()*. Selanjutnya, ketika MIDlet dijalankan, MIDlet akan berada pada *state* "*Active*", dalam hal ini fungsi standar yang dipanggil adalah *startApp()*. Akan tetapi, jika ditengah jalan MIDlet dihentikan sementara, MIDlet akan berada dalam *state* "*Paused*" dengan memanggil fungsi standar *pauseApp()*. Pada state ini diperlukan proses *cleanup* terhadap *garbage collector* yang dihasilkan.

User interface dari MIDP terdiri dari *Low Level* dan *High Level API*. *Low Level API* berbasis pada *class Canvas* sedangkan *High Level API* berbasis pada *Screen*. Contoh dari *High Level API* adalah *Alert*, *Form*, *List*, *TextBox*.

Dalam pembuatan suatu aplikasi MIDlet, aplikasi yang dibuat harus dikemas dalam sebuah file berekstensi *.JAR. Selain itu dikenal juga dikenal juga file *.JAD yang disebut sebagai *application descriptor* yang berfungsi untuk mendeskripsikan dari file JAR tersebut. Kedua file ini harus di-*upload* ke ponsel pengguna agar aplikasi dapat dijalankan.

2.15.1. Batasan MIDlet

Karena Java bekerja dalam lingkungan yang dibatasi oleh Java Virtual Machine, maka MIDlet tidak memiliki kemampuan untuk mengakses file system yang terdapat pada handset pengguna. Berikut adalah diagram yang menggambarkan kedudukan MIDlet pada perangkat pendukung-nya :



Gamabar 2.4 Diagram MIDlet

Secara umum dapat dikatakan bahwa MIDlet tidak dapat mengakses seluruh kemampuan handset pengguna sebagaimana yang dapat dila-kukan oleh aplikasi *native*. Meskipun demikian dengan menggunakan class spesifik OEM, maka MIDlet dapat menggunakan fitur-fitur yang terdapat pada handset.

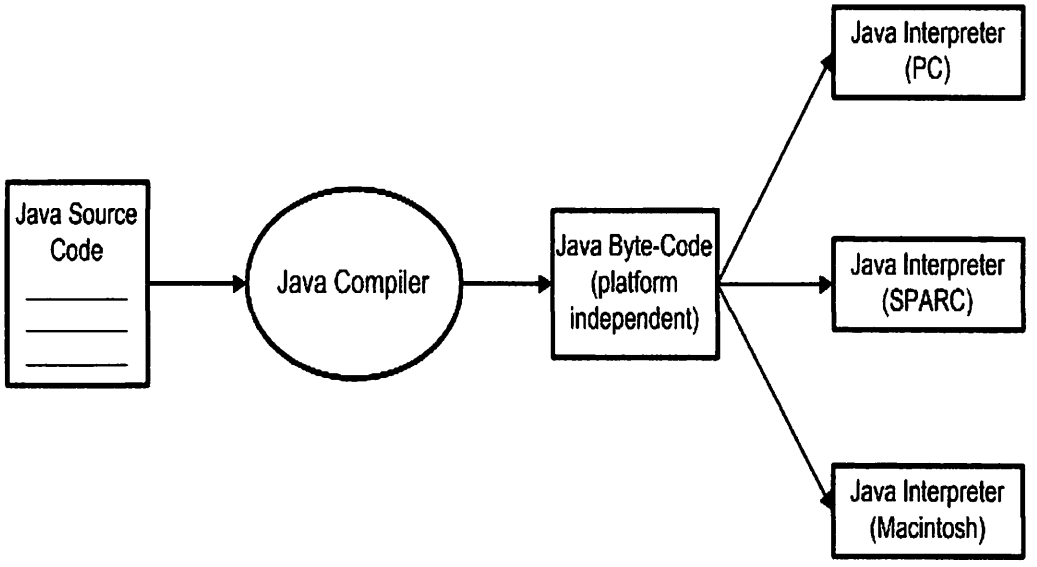
Sebagai contoh MIDP 1.0 tidak menspesifikasikan fasilitas membangkit-kan suara. Tapi dengan menggunakan spesifikasi milik Nokia misalnya, maka dimungkinkan untuk membuat MIDlet yang dapat membangkitkan suara untuk semua handset Nokia. Demikian pula dengan vendor-vendor lainnya seperti Motorola dan Siemens.

2.15.2. Siklus Hidup MIDlet

Dalam eksekusinya, suatu aplikasi MIDlet akan mempunyai beberapa *fase* atau *state* yang menunjukkan keadaannya pada saati itu. Perubah-an *fase* atau *state* ini disebut sebagai *LifeCycle* pada MIDlet. *LifeCycle* dari sebuah MIDlet ditangani oleh AMS (*Application Manage-ment System*) atau JAM (*Java Application Manager*). AMS adalah ling-kungan tempat siklus dari sebuah MIDlet diciptakan, dijalankan, dihenti-kan maupun dihilangkan.

MIDlet mempunyai tiga *state*, yaitu **PAUSE**, **ACTIVE** dan **DESTROY**. *State* **PAUSE** terjadi ketika MIDlet tidak melakukan aksi apapun, *state* **ACTIVE** terjadi ketika MIDlet sedang aktif atau berjalan, sedangkan *state* **DESTROY** terjadi ketika MIDlet berhenti berjalan dan alokasi memori yang digunakan oleh MIDlet akan dibersihkan. Ketika masing-masing state dipanggil, beberapa method yang bersesuaian dipanggil.

Ketika AMS pertama kali menciptakan dan menginisialisasi MIDlet, MIDlet akan berada dalam *state* **PAUSE**. Apabila terjadi kesalahan selama konstruksi MIDlet, MIDlet akan berpindah ke *state* **DESTROY** dan MIDlet batal diciptakan dengan jalan memanggil fungsi standar destroyApp(). Selanjutnya ketika MIDlet dijalankan, MIDlet akan berada pada *state* **ACTIVE** dengan jalan memanggil fungsi standar startApp(). Tetapi ketika ditengah jalan MIDlet dihentikan sementara, MIDlet berada pada *state* **PAUSE** dengan memanggil fungsi standar pauseApp(). Pada state ini juga dilakukan proses cleanup pada garbage colector yang dihasilkan.



Gambar 2.5 Siklus Hidup MIDlet

Suatu aplikasi MIDlet terdiri atas satu buah turunan class MIDlet beserta satu atau lebih turunan class Displayable yang bertindak sebagai *user-interface*. Karena class MIDlet mengandung method abstract, maka untuk membuat sebuah turunan class MIDlet kita harus mendefinisikan terlebih dahulu method-method abstract yang dimiliki oleh class MIDlet tersebut dalam class yang akan dibuat nantinya. Method tersebut adalah :

startApp()	Akan dipanggil saat MIDlet memasuki fase "started" dan fase "paused". Disini biasanya dilakukan proses inisialisasi, baik inisialisasi layar maupun objek-objek yang dipergu-nakan dalam aplikasi MIDlet.
pauseApp()	Akan dipanggil saat MIDlet memasuki fase "paused" Biasanya berisi instruksi untuk membebaskan me-mory yang dipakai oleh aplikasi MIDlet, sekaligus menyimpan kondisi aplikasi MIDlet saat ini.
destroyApp()	Akan dipanggil saat MIDlet memasuki fase "destroy-ed" yaitu ketika eksekusi MIDle dihentikan.

Dengan demikian, maka kerangka program dari turunan class MIDlet yang akan dibuat, dapat ditulis sebagai berikut :

```
1   import javax.microedition.midlet.*;
2
3   public class ClassMidlet extends MIDlet
4   {
5       public ClassMidlet()
6       {
7         //-- konstruktor MIDlet
8       }
9
10      public void startApp()
11      {
12        //-- kode ketika MIDlet memasuki fase ACTIVE
13      }
14
15      public void pauseApp()
16      {
17        //-- kode ketika MIDlet memasuki fase PAUSED
18      }
19
20      public void destroyApp()
21      {
22        //-- kode ketika MIDlet berakhir...
23      }
24  }
```

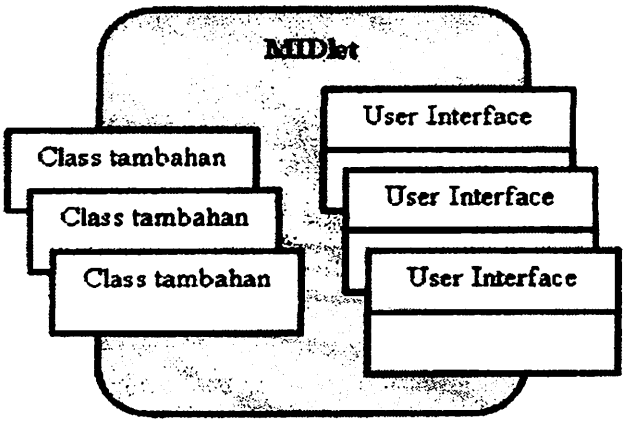
Kode pada baris pertama dipergunakan untuk memberitahukan kompi-lator Java bahwa program ini menggunakan semua class yang terdapat pada *package* java.microedition.midlet. Pada *package* inilah terdapat definisi class MIDlet dan interface lain yang diperlukan.

Kode pada baris ketiga merupakan definisi dari class yang diturunkan dari class MIDlet. Perlu diperhatikan bahwa class baru ini harus didefinisikan dengan modifier public. Jika tidak maka perangkat MID tidak dapat mengeksekusi MIDlet.

Kode pada baris kelima merupakan definisi constructor dari class baru tersebut. Nama dari constructor sama dengan nama classnya.

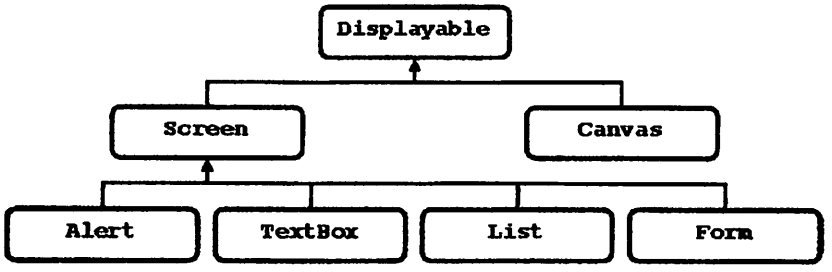
2.15.3. Struktur MIDlet

Suatu aplikasi MIDlet disusun atas satu buah *class* utama yang diturun-kan dari *class* MIDlet. *Class* ini adalah *class* yang pertama kali dijalan-kan oleh perangkat handset ketika aplikasi MIDlet dieksekusi. Karena *class* MIDlet tidak menyediakan *user interface* apapun, maka untuk membuat *user interface* dipergunakan *class-class* yang diturunkan dari *class* Displayable. Dengan melalui *class-class* inilah aplikasi dapat berinteraksi dengan pengguna.

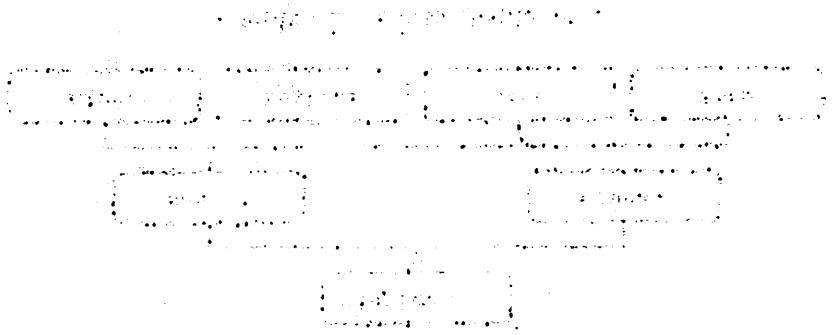


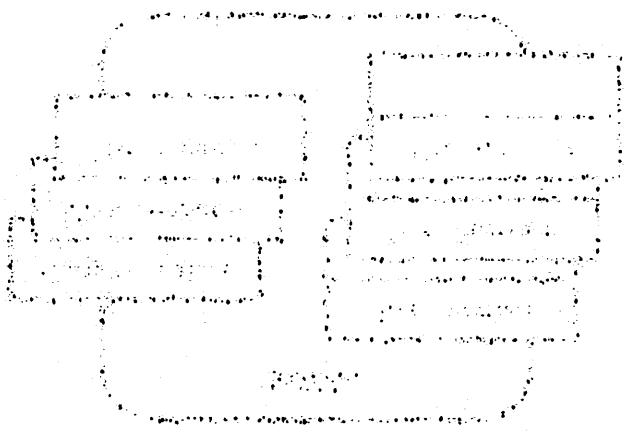
Gambar 2.6 MIDlet Class

Class Displayable itu sendiri adalah *class* abstract, yang diperlukan pada suatu aplikasi MIDlet adalah turunan dari *class* tersebut. Pada MIDP, *user interface* terbagi menjadi dua, yaitu *High Level API* dan *Low Level API.*



Gambar 2.7 Class Displayable





High Level API menyediakan *class-class* yang mekanismenya telah ditentukan dan dapat langsung dipergunakan dalam pembuatan aplikasi MIDlet. Programmer hanya mendefinikan "isi" dari *user interface* tersebut, menampilkan kepada pengguna dan kemudian mendapatkan hasilnya. *Class* yang dipergunakan adalah *class* yang diturunkan dari class Screen.

Sedangkan *Low Level API* merupakan media yang dipergunakan dalam aplikasi yang melakukan operasi grafis, seperti game misalnya. Disini programmer bertanggung jawab dalam menentukan tampilan dan mekanisme pada layar aplikasi. *Class* yang dipergunakan adalah *class* yang diturunkan dari *class* Canvas.

Selain *class* untuk *user interface*, dalam suatu aplikasi MIDlet dapat pula mempunyai *class-class* lainnya, seperti *class* yang bertugas untuk melakukan proses tertentu dalam aplikasi tersebut.

2.16 JDK (Open JDK)

Menurut Charibaldi, Novrido (2010, pp2-4), untuk membuat program di Java, kita terlebih dahulu harus memiliki JDK (*Java Development Kit*). JDK ini berisi *compiler* dan JVM (*Ja va Virtual Machine*). Sebelum program Java dijalankan, terlebih dahulu *compiler* yang akan bekerja unt uk mengkompilasi pro gram Java menjadi *bytecod e*. Kemudian *bytecode* inilah yang nantinya akan dijalankan oleh JVM.

2.17 Java IDE

Menurut Charibaldi, Novrido (2010, p4), IDE (*Integrated Development Environment*) merupakan *tools* pembantu untuk pembuatan program. Berbeda dengan Visual Basic atau bahasa pemrograman yang lain dimana pengguna tidak diperbolehkan memilih IDE-nya sendiri. Java membebaskan penggunanya untuk memilih IDE yang mereka sukai.

Beberapa IDE Java yang paling popular antara lain sebagai berikut:

1. Netbeans
2. Eclipse
3. Jdeveloper

2.18 Kelebihan Dan Kekurangan Java

A. Keunggulan Java :

   1. Berorientasi objek

Java menggunakan konsep pemrograman berorientasi objek (OOP). OOP memungkinkan adanya *reuseable code* dan komunikasi antar objek.

2. *Platform independent*

   Java merupakan bahasa yang tidak terikat dengan platform manapun(Windows, Linux, Mac, dll). Cukup buat sekali bisa kita jalankan dimanapun (*Write Once Run Anywhere*).

3. *Robust*

   Compiler Java memiliki kemampuan mendeteksi kesalahan lebih teliti dibandingkan bahasa pemrograman yan g lain.

4. *Interpreter*

   Java menggunakan *interpreter* berupa JVM. JVM inilah yang akan mengkompilasi kode Java menjadi *bytecode* dan dapat dijalankan di platform yang berbeda.

5. Aman

   Java memiliki tingkat keamanan yang tinggi untuk menjaga supaya aplikasi tidak merusak sistem komputer yang menjalankannya.

6. Sederhana

   Java mengadopsi sintaks bahasa yang mirip dengan bahasa C/C++, yang menjadikan Java menjadi mudah untuk dipahami. Dan melakukan penyempurnaan dengan menghilangkan *pointer* yang selalu dianggap bermasalah.

7. Dukungan komunitas

   Java memiliki komunitas yang sangat banyak diseluruh penjuru dunia. Komunitas Java yang paling terkenal adalah JUG (*Java User Groups*). Selain komunitas, juga terdapat banyak sekali forum Java di Internet yang disediakan untuk membantu para penggemar Java yang mengalami kesulitan.

8. Teknologi sekaligus solusi

   Java merupakan sebuah bahasa yang lengkap (*One fo r All*). Satu buah bahasa tetapi memiliki teknologi yang dapat digunakan untuk lingk un gan aplikasi yang berbeda. Selain sebuah bahasa Java juga merupakan sebuah solusi *open source*. Sudah banyak solusi yang dikembangkan menggunakan Java dan bersifat *open source*, antara lain: alfresco(*File Server*), liferay (portal), adempiere (ERP).

9. Kecepatan yang sudah diperbaiki

   Java bukan sebuah bahasa yang baru saja keluar 1 atau 2 tahun yang lalu, tetapi Java sampai saat in i sudah berusia kurang lebih 13 tahun. Java sudah san gat *mature*. Kecepatan Java sudah jauh lebih baik dari pada saat Java pertama kali muncul. Ditambah lagi den gan harga *hardware computer* yang sudah mulai terjangkau.

10. Akses tanpa koneksi dan sinkronisasi (*Disconnected Access and Synchronization*)

    Aplikasi *mobile* yang menggunakan teknologi Java dapat berjalan bahkan ketika sebuah perangkat sedang tidak terkoneksi atau berada di luar *coverage area*. Dengan menggunakan aplikasi berbasis teknologi Java, *user* dapat menjalankan dan berinteraksi dengan sebagian besar aplikasi pada telepon selular dan perangkat bergerak lainnya pada *standalone mode* dan pada nantinya bersinkronisasi dengan infrastruktur *back-end*. Hal ini bertentangan dengan perangkat yang menggunakan WAP, yaitu *user* harus terkoneksi setiap waktu. Jika *user* berada di luar *coverage area*, maka aplikasi yang memerlukan layanan WAP tidak dapat digunakan.

11. Penyampaian aplikasi dan layanan secara dinamis (*Dynamic delivery of applications and services*)

    Tidak seperti perangkat-perangkat lainnya pada saat skripsi ini ditulis, telepon selular dan PDA generasi mendatang akan memiliki kemampuan untuk men-*download* aplikasi secara aman dan *realtime*. Teknologi Java memungkinkan *user* men-*download* aplikasi baru secara dinamis dan kemudian menjalankannya pada perangkat yang dimiliki *user*.

12. Kompatibilitas antar platfom (*Cross-platform compatibility*)

    Karena aplikasi-aplikasi yang dibuat dengan menggunakan teknologi Java dapat berjalan pada banyak jenis perangkat, aplikasi yang sama juga dapat dijalankan pada PDA maupun telepon selular.

13. *Enchaned user experience*

    Karena aplikasi-aplikasi yang berbasiskan teknologi Java memiliki tampilan yang lebih menarik dengan interaksi yang lebih cepat, *developer* dapat membuat aplikasi yang lebih menarik dan berguna jika dibandingkan dengan *environment* berbasis *browser* yang sudah ada,

seperti WAP. Kebutuhan *bandwidth* jaringan juga berkurang karena tampilan grafis dihasilkan secara *local.*

14. Skalabilitas dan performa (*Scalability and Performance*)

    Penggunaan teknologi Java pada *client end* memungkinkan *user* bekerja secara *offline* (*standalone mode*) yang berdampak pada berkurangnya *user* yang mengakses *server* pada waktu tertentu. Hal ini juga berdampak pada meningkatnya performa dan skalabilitas (dapat digunakan dalam sebuah rentang kemampuan) *server* dan mengurangi kebutuhan akan *bandwidth* jaringan.

15. Mendukung *Garbage Collector*

    Java memiliki fasilitas *garbage collector* yang berarti java akan menghapus secara otomatis objek-objek yang sudah tidak dibutuhkan lagi. Fasilitas ini dapat mengurangi beban terhadap pengelolaan *memory* oleh *programmer.*

B. Kekurangan Java

Proses Compile, Mengharuskan pengguna mengcompile programnya sebelum dijalankan, berbeda dengan bahasa pemrograman python yang tidak perlu mengcompile terlebih dahulu. Penggunaan Memori yang besar , Berbeda dengan bahasa pemrograman lain yang hanya membutuhkan memori sedikit

BAB III
ANALISA DAN PERANCANGAN SISTEM

Pada bab ini dijelaskan mengenai analisis dan perancangan sistem aplikasi. Analisis ditujukan untuk memberikan gambaran secara umum dari aplikasi dan memberikan solusi terhadap permasalahan yang dihadapi. Hal ini berguna untuk menunjang perancangan aplikasi yang akan dikembangkan sehingga kebutuhan akan aplikasi tersebut dapat diketahui sebelumnya. Kemudian hasil analisis akan menjadi dasar untuk melakukan perancangan atau desain aplikasi sesuai kebutuhan sistem.

Dalam merancang aplikasi pada projek ini terlebih dahulu dilakukan pembuatan desain sistem, desain proses, serta desain antar muka aplikasi. Desain proses berguna untuk mengintegrasikan semua proses yang terjadi dalam aplikasi yang akan dibuat. Desain data berguna untuk mengetahui apa saja yang dibutuhkan dalam proses yang akan dikerjakan. Sedangkan perancangan antarmuka berfungsi sebagai antar muka interaksi antara pengguna dengan sistem aplikasi yang dibuat, sehingga pengguna dapat mengoperasikan aplikasi yang dibuat.

Dalam menyelesaikan masalah diatas digunakan metode dalam meneliti masalah yang dihadapi. Adapun metode penelitian yang digunakan adalah :

3.1. Tujuan Pembuatan Sistem

Perancangan dan pembuatan sistem merupakan hal yang sangat penting dalam proyek akhir ini. Sebagai langkah awal dalam pembuatan sistem dibutuhkan suatu perencanaan terhadap segala komponen-komponen yang diperlukan dalam pembuatan sistem, karena dengan perencanaan tersebut diharapkan nantinya akan mendapatkan suatu sistem yang baik dan siap untuk dioperasikan dengan yang diharapkan.

Tujuan pembuatan sistem aplikasi email client ini adalah membuat suatu aplikasi yang dapat menerima dan mengirim pesan elektrik secara realtime maupun dengan hanya dengan metode *push mail* dengan mempergunakan handphone.

3.2. Perencanaan Sistem

Perencanaan sistem yang akan dibuat adalah mengenai proses pengiriman pesan elektrik dari server *email* pada hanphone nokia sebagai tujuan untuk menerima pesan elektrik. Untuk melakukan proses pengiriman pesan elektrik dibutuhkan *settings* pada

BAB III
ANALISA DAN PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai analisa dan perancangan sistem aplikasi. Analisis dilakukan untuk mendefinisikan kebutuhan yang diperlukan dari sistem aplikasi yang akan dibangun. Perancangan sistem bertujuan untuk memberikan gambaran kepada pengembang perangkat lunak dan pengguna aplikasi yang akan dibangun. Hal ini bertujuan untuk mempermudah pengembang aplikasi yang akan dibangun sehingga kebutuhan akan aplikasi tersebut sesuai dengan spesifikasi. Kemudian hasil analisis akan menjadi dasar dilakukan perancangan sistem dengan aplikasi sesuai kebutuhan pengguna.

Proses perancangan aplikasi pada penelitian ini menggunakan metode pengembangan sistem yang akan dibangun akan dijelaskan. Dengan proses berupa metode pengembangan sistem perangkat lunak yang akan digunakan dalam proses pembuatan aplikasi. Kemudian dijelaskan juga cara pengumpulan data yang akan digunakan dalam proses pembuatan aplikasi. Dengan adanya metode pengembangan sistem aplikasi yang akan dibangun pengguna dapat menggunakan aplikasi yang dibuat.

Dalam pengembangan metode akan digunakan beberapa metode yang akan digunakan. Adapun metode penelitian yang digunakan adalah :

3.1. Tujuan Pembentukan Sistem

Perancangan alur pembangunan sistem menggunakan hal yang sangat penting dalam proses akhir sebuah laporan. Hal dalam pembuatan alur sistem diharapkan sistem pembangunan berjalan sesuai komponen-komponen yang digunakan dalam pembuatan sistem, karena dengan perancangan sistem dapat mempermudah dalam pembuatan sistem yang baik dan untuk dikembangkan dengan yang akan datang.

Tujuan pembentukan sistem aplikasi untuk alur sistem ini adalah membuat sistem dengan baru dengan menambah alur sistem sebelum sistem melakukan proses pembuatan sistem yang akan dibangun.

3.2. Perancangan Sistem

Perancangan sistem yang akan dibuat adalah rancangan proses pembuatan pusat elektrik dan sensor yang terdiri dari rancangan aplikasi untuk memudahkan pengguna

email server dan email client sehingga pesan dapat diterima dan dikirim melalui *hanphone*.

3.3 ANALISA SISTEM

Metode penelitian yang digunakan berupa simulasi proses pengiriman dan penerimaan *e-mail* pada *handphone* dengan membuat *emulator* dari aplikasi yang ada pada *handphone*.

Beberapa perangkat yang dibutuhkan unntuk menjalankan aplikasi email client via mobile berbasis symbian ini adalah.

1. **Perangkat Keras *(Hardware)***
   - *a. Satu Unit Personal Computer(PC)*
   - *b. Satu Unit Printer*
   - *c. Satu Unit Handset Smartphone Nokia E52*
   - *d. Flashdisk untuk pemindahan data*

2. **Perangkat Lunak *(Software)***
   - a. Sistem operasi : Linux Debian Wheezy
   - b. *Tools* Perancangan Sistem : Dia
   - c. IDE (*Integrated Development Apllication) : Netbeans*
   - *d. Wireless Tolkit 2.5.2 ml-linux*

3.4. Data-Data yang Diperlukan

Dalam implementasi sistem untuk memperoleh serta memproses suatu informasi ini ada beberapa langkah yang harus dilakukan antara lain :

1. Analisis

   Tahap ini melakukan pengumpulan elemen-elemen yang berkaitan dengan proses implementasi dan data yang diperlukan, meliputi: data, metode, fungsi, proses atau procedure. Hasil akhir tahap ini adalah spesifikasi kebutuhan yang diperlukan, khususnya dalam hal perangkat lunak.

[illegible]

3.2 [illegible]

[illegible]

[illegible]

1. [illegible]
   a. [illegible]
   b. [illegible]
   c. [illegible]
   d. [illegible]

2. [illegible]
   a. [illegible]
   b. [illegible]
   c. [illegible]
   d. [illegible]

3.2.1 Data-flow [illegible]

[illegible]

1. [illegible]

   [illegible]

2. Design

   Spesifikasi perangkat lunak yang dihasilkan dari tahap analisa ditransformasikan kedalam bentuk arsitektur perangkat lunak yang memiliki karakteristik mudah dimengerti dan tidak sulit untuk diimplementasikan.

3. Coding / Pemrograman

   Tahap ini dilakukan implementasi hasil rancangan kedalam baris-baris kode program yang dapat dimengerti oleh mesin.

4. Testing

   Pengujian dilakukan untuk setiap modul. Jika hasil pengujian tidak menemukan adanya masalah, modul-modul yang terpisah tersebut diintegrasikan untuk mendapatkan perangkat lunak yang utuh. Kemudian, dilakukan pengujian ditingkat perangkat lunak yang memfokuskan pada masalah-masalah logika internal, fungsi eksternal, potensi masalah yang mungkin terjadi dan pemeriksaan hasil.

3.5 Desain Sistem

Design Sistem merupakan penggambaran perencanaan dan pembuatan sketsa dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan mempunyai fungsi. Untuk memberiakan gambaran yang jelasa dan rancang bangun yang lengkap dari desain aplikasi aplikasi *email client* ini.

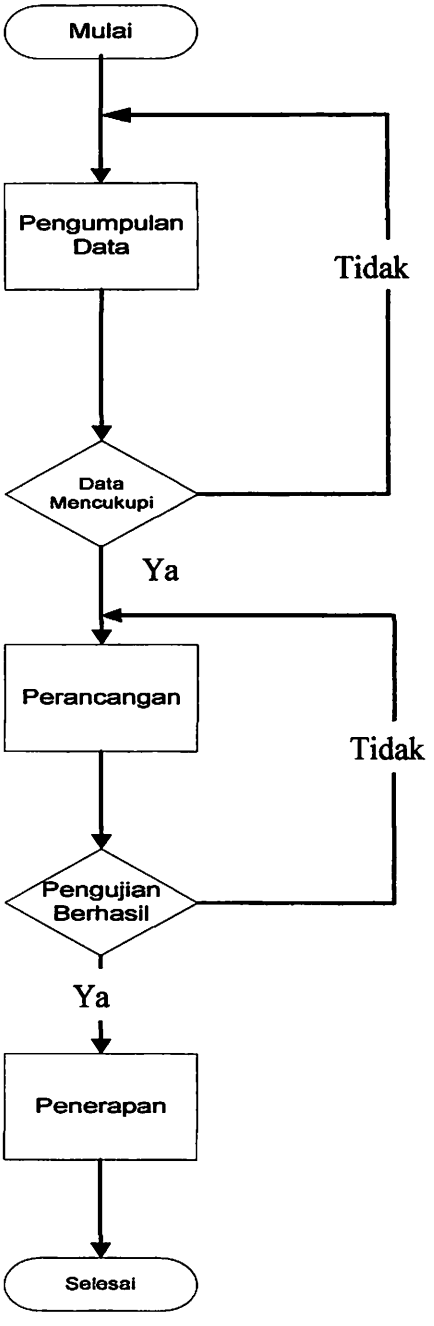
3.6 Perancangan Sistem Aplikasi

Tujuan utama dari perancangan sistem adalah memberiakan gambaran perancangan sistem yang akan dibangun atau dikembangkan, serta memahami alur aplikasi dan proses dalam sistem.

3.6.1 Rancang Diagram ALir

Perancangan diagram alir akan menjelaskan bagaimana proses dan urutan dari aplikasi yang dibuat. Bagian-bagian perencanaan diagram alir tersesbut diantaranya yaitu diagram alir diagram alir *email client.*Dengan perencanaan,kita dapat menyimpulkan dimana proses-proses untuk mengirim serta menerima *e-mail.*

3.6.2 Flowchart

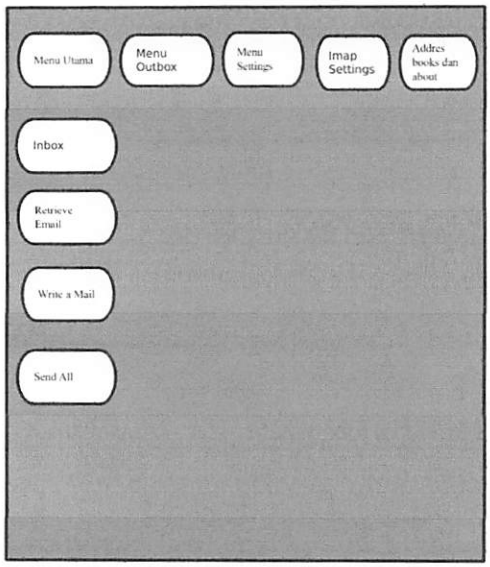


Gambar 1.1. Diagram Alir Penelitian

3.6.3 Perancangan Interface

Aplikasi ini beroperasi pada ponsel yang memiliki keterbatasan memori, sumber daya, dan kemampuan memproses. Penggunaan *Java Micro Edition* (JME) untuk mengembangkan aplikasi *email client* untuk memproses informasi perkuliahan karena JME didesain untuk meletakkan perangkat lunak java pada perangkat elektronik, seperti

ponsel. Selain alasan tersebut penggunaan JME karena hampir semua ponsel yang ada sekarang sudah mendukung penggunaan bahasa pemrograman java ini. Pada pengembangan aplikasi ini *Configuration* yang digunakan adalah CLDC (*Connected Limited Device Configuration*) versi 1.0 dan *profile* yang digunakan MIDP (*Mobile Information Device Profile*) versi 2.0. Untuk *widget* UI menggunakan LWUIT (*Lightweight User Interface Toolkit*).

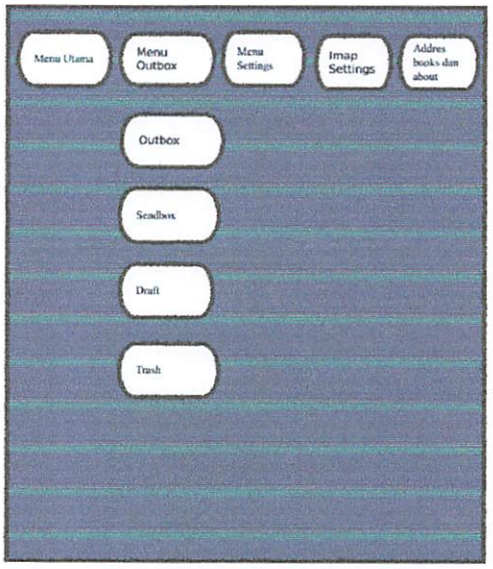


Gambar 3.1 Tampilan Menu Utama

Gambar 3.2 Tampilan Inbox pada Netbeans

Keterangan Gambar:

Pada menu Utama atau Menu Inbox berjalan baik dengan menampilkan menu kategori Inbox, Retrieve Email, Write Mail, Send All. Untuk lebih jelasnya pada Gambar 3.1 dan Gambar 3.2.

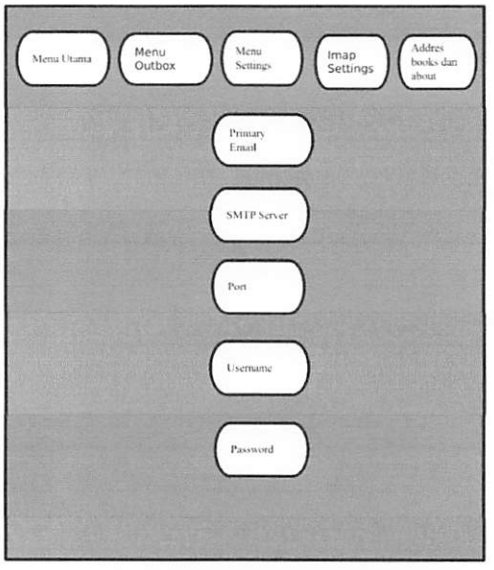


Gambar 3.3 Tampilan Menu Outbox

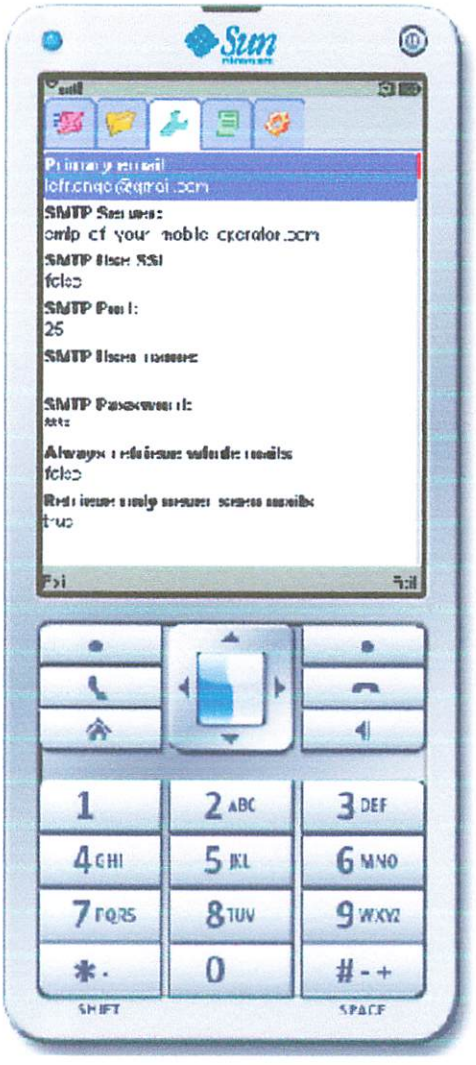
Gambar 3.4 Tampilan Outbox pada Netbeans

Keterangan Gambar:

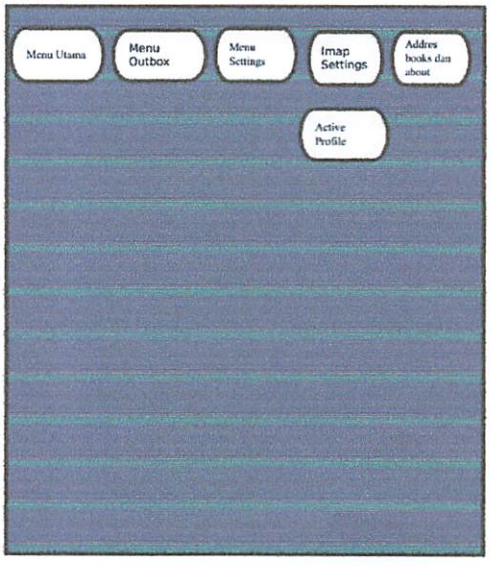
Pada menu *Outbox* berjalan baik dengan menampilkan menu kategori *Outbox, Senbox, Draft, Trash*. Untuk lebih jelasnya pada Gambar 3.3 dan Gambar 3.4.



Gambar 3.5 Menu Settings



Gambar 3.6 Tampilan pada Netbeans

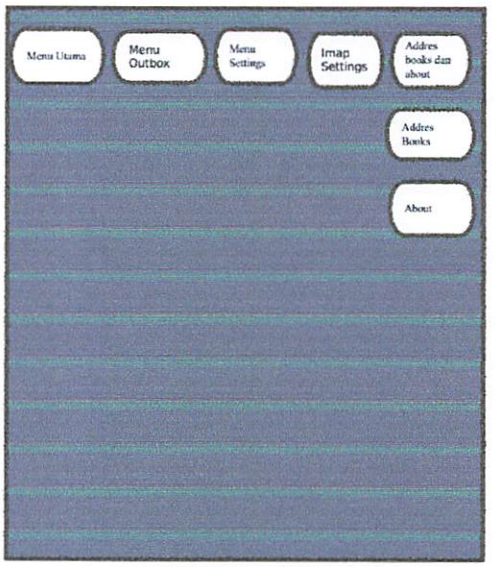


Gambarv3.7 Imap Settings

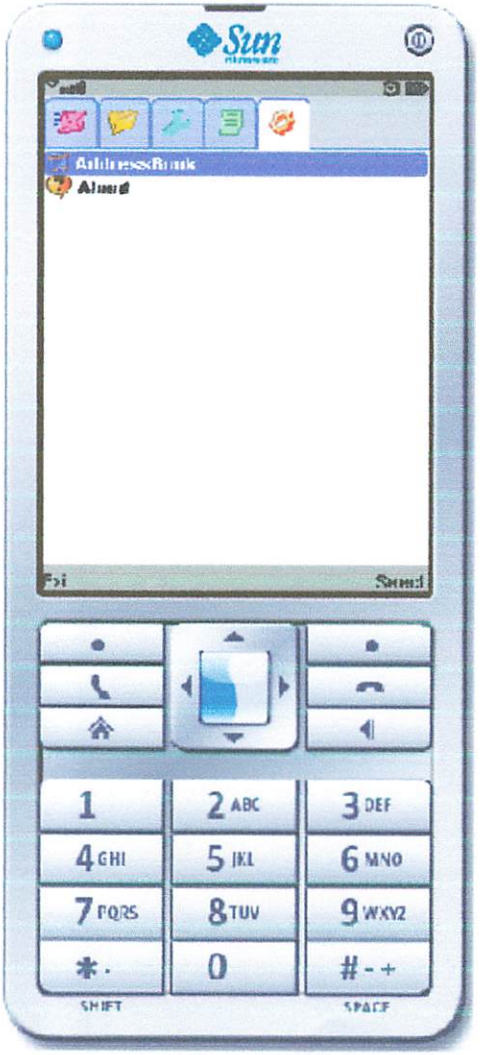
Gambar 3.8 Tampilan pada Netbeans

Keterangan Gambar:

Pada menu Settings berjalan baik dengan menampilkan menu kategori Setting IMAP Server, SMTP Server, Port, dan settings lainnya. Untuk lebih jelasnya pada Gambar 3.5, Gambar 3.6 Gambar 3.7 dan Gambar 3.8



Gambar 3.9 Menu Addres Books and About



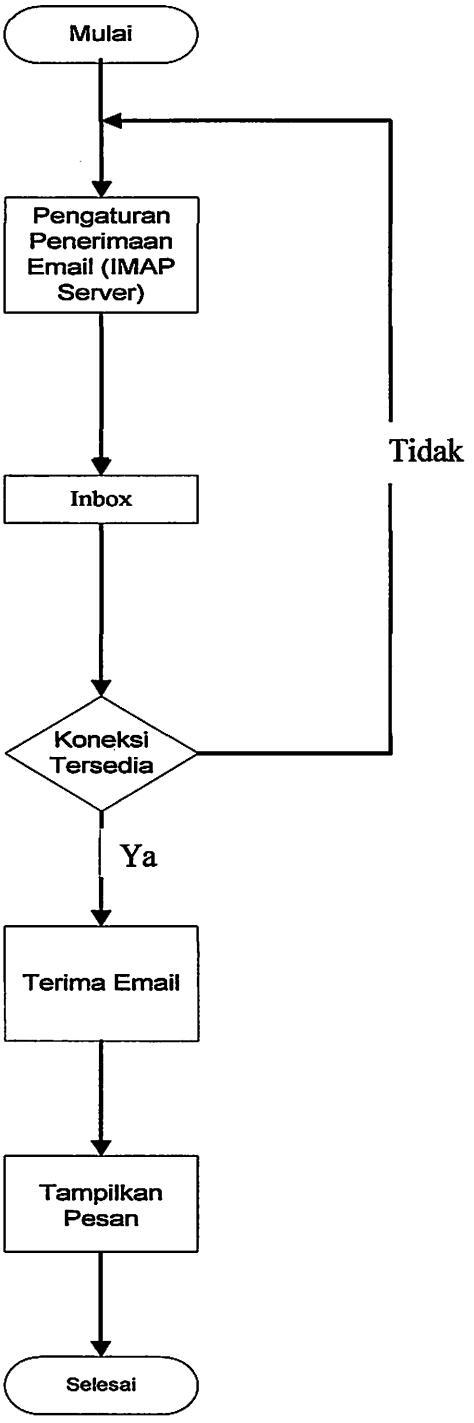
Gmabar 3.10 Tampilan pada Netbeans

Keterangan Gambar:

Pada menu History berjalan baik dengan menampilkan menu kategori *addres books dan about*. Untuk lebih jelasnya pada Gambar 3.9 dan Gambar 3.10

3.7 Blok Diagram Jalannya Program

Jalannya program secara garis besar yang didesain untuk aplikasi penerimaan email terdiri dari beberapa langkah. Berikut ini blok diagram penerimaan dan penampilan pesan elektronik, ditunjukkan dalam gambar 3.3 dibawah ini :



3.3 Flowchat Alur Aplikasi

BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bab ini akan dijelaskan mengenai implementasi dan uji coba dari Aplikasi *E-Mail Java Mobile Client* ini.

4. PEMBUATAN DAN PENGUJIAN SISTEM

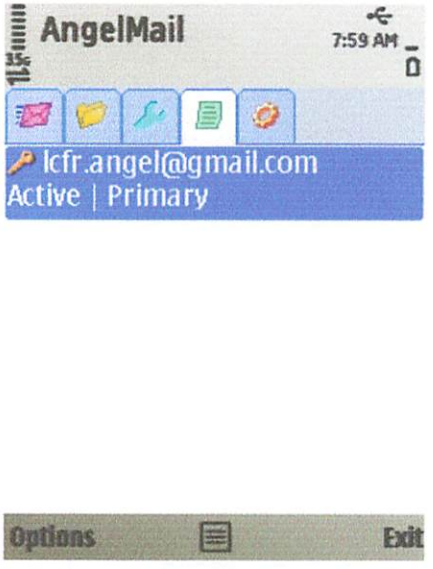
4.1 Implementasi Sistem

Tahap implementasi pengembangan perangkat lunak merupakan proses pengubahan spesifikasi sistem menjadi sistem yang dapat dijalankan. Tahap ini merupakan lanjutan dari proses perancangan, yaitu proses pemrograman perangkat lunak sesuai dengan spesifikasi dan desain sistem.

4.2 Pengujian Aplikasi

Pada pengujian Aplikasi Email Client ini akan dilakukan dengan *handphone* berbasis Symbian Java yaitu, Nokia E52. Dalam pengujian ini untuk mengetahui tingkat keberhasilan dari aplikasi ini

4.2.1 Halaman Profile

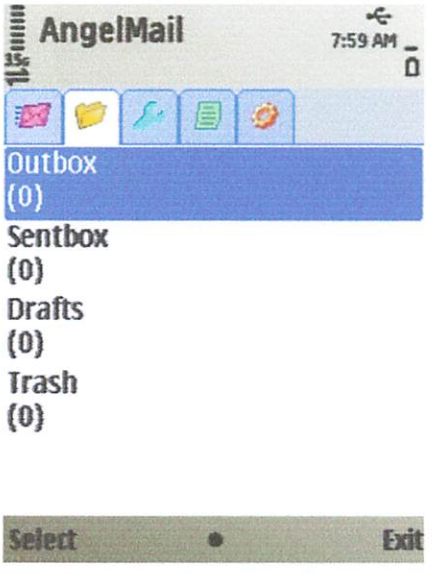


Gambar 4.1 Tampilan Halaman Profile

4.2.2 Halaman Settings

Gambar 4.2 Menu Tampilan Settings

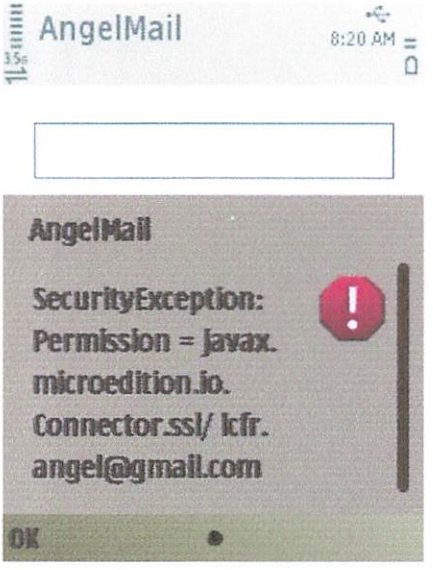
4.2.3 Tampilan Halaman Inbox,*Sendbox, Draft, Trash*



Gambar 4.3 Halaman Inbox *Sendbox, Draft, Trash*

4.2.4 Tampilan Menu Koneksi

Gambar 4.4 Menu Koneksi



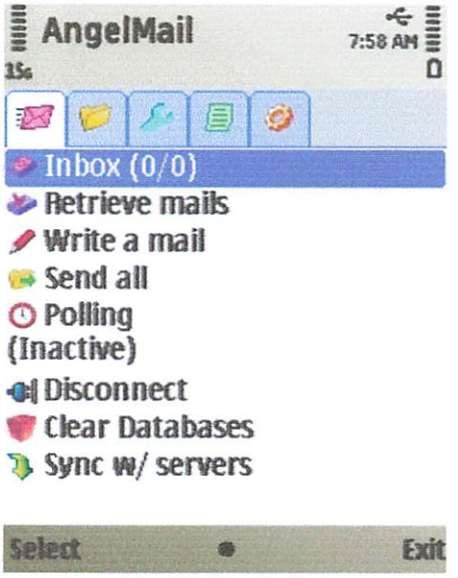
Gambar 4.5 Kegagalan dalam koneksi

4.3 Pengujian Hasil

Setelah pembuatan rancang bangun, dihasilkan suatu aplikasi *email client*, dimana setelah dilakukan pengujian pada program ini melalui prosedur - prosedur dalam bahasa pemrograman Java, program ini dapat berjalan dengan baik. Pengujian hasil difokuskan pada pengujian aplikasi.

4.3.1 Pengujian Menu Utama

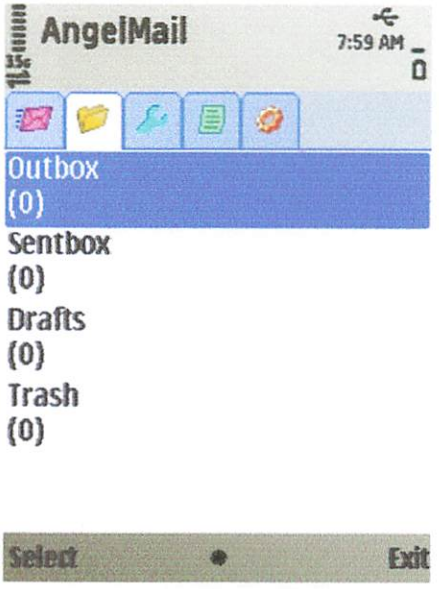
Pada pengujian menu utama, aplikasi dapat berjalan dengan baik dengan menampilkan menu kategori Inbox, Retrieve Email, Write Mail, Send Mail. Untuk lebih jelasnya pada gambar 4.6:



Gambar 4.6 Pengujian Menu Utama

4.3.2 Pengujian Menu Outbox

Pada pengujian menu outbox, aplikasi ini dapat berjalan dengan baik dengan menampilkan outbox, sendbox, draft, trash. Untuk lebih jelasnya pada gambar 4.7 :



Gambar 4.7 Pengujian Menu Outbox menampilkan outbox, sendbox, draft, trash

4.3.3 Pengujian Menu Settings

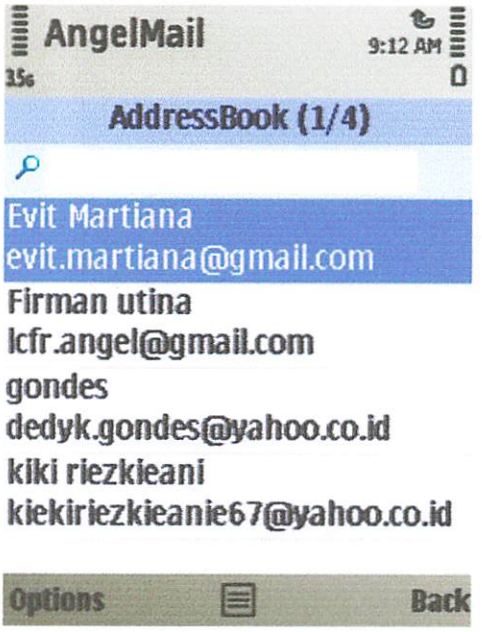
Dalam pengujian menu settings, aplikasi dapat berjalan dengan baik dengan menampilkan berbagai *settings* mulai dari setting smtp server, *settngs port*. Untuk lebih jelasnya pada gambar 4.8, setting imap server 4.9:

Gambar 4.8 Pengujian *Menu Settings menampilkan smtp server, settngs port*

Gambar 4.9 Pengujian *Menu Settings menampilkan imap server*

4.3.5 Pengujian Menu Addres Books dan About

Dalam pengujian menu address boooks dan about, aplikasi dapat berjalan dengan baik dengan menampilkan tentang riwayat Aplikasi *email client ini*. Untuk lebih jelasnya pada gambar 4.10:



Gambar 4.12 Pengujian Menu About dan Contacts

BAB V
PENUTUP

5. PENUTUP

5.1 Kesimpulan

Dengan dibuatnya Aplikasi Email Client menggunakan koneksi internet ini , dapat disimpulkan, para pengguna mendapatkan informasi tentang :

a. Memudahkan pengguna dalam melihat isi email tanpa menggunakan desktop machine.

b. Pengguna dapat sewaktu waktu menerima email dimanapun berada, bila tersedia koneksi internet.

c. Mendukung koneksi SSL (Secure Socket Layer)

d. Aplikasi dapat ditanam pada ponsel yang sudah mendukung aplikasi java yang minimal memiliki CLDC (*Connected Limited Device Configuration*) versi 1.0 dan MIDP (*Mobile Information Device Profile*) versi 2.0.

e. Media yang digunakan untuk penyampaian informasi mengunakan *email.*

f. *Setting* akun *email* dalam aplikasi hanya ditujukan untuk satu akun *email* saja. namun demikian aplikasi dapat menyimpan beberapa *profile* akun *email* yang dapat digunakan untuk merubah *setting* akun *email* tersebut.

g. *Email* diperoleh dengan cara menarik (*pull*) *email* dari *mail server* dengan interval waktu yang telah ditentukan.

5.2 Saran

Aplikasi email menggunakan koneksi internet ini masih dapat dikembangkan lebih jauh lagi karena dalam pembuatannya masih jauh dari sempurna dan masih banyak kekurangan. Adapun saran yang dapat dikemukakan agar aplikasi ini bisa berfungsi dengan lebih optimal adalah:

1. Untuk saat ini aplikasi hanya bisa memproses satu akun *email* dalam satu waktu. Diharapkan dalam pengembangan aplikasi ini kedepannya bisa memproses beberapa akun *email* sekaligus.

3. Pengiriman *email* ke akun *email* pengguna aplikasi *email client* ini masih dilakukan secara *manual*, sehingga dibutuhkan aplikasi *server* untuk mengatur pengiriman *email* tersebut, sehingga kedepannya diharapkan dapat menerima dan mengirim pesan elekrik secara real time.

DAFTAR PUSTAKA

[1] Shalahuddin, M., *Pemrograman J2ME Belajar Cepat Pemrograman Perangkat Telekomunikasi Mobile,* Informatika Bandung, Bandung. 2006

[2] Wicaksono, A., *Pemrograman Internet dan XML pada ponsel dengan MIDlet Java,* PT Elex Media Komputindo, Jakarta. 2003

[3] Wicaksono, A., *Pemrograman Aplikasi Wireless dengan Java,* PT Elex Media Komputindo, Jakarta. 2002

[4] Yuan, M.Y., *Enterprise J2ME: Developing Mobile Java Applications,* Prentice Hall PTR. 2003

[5] Microsystem, Sun, *JavaME,* http://www.oracle.com/technetwork/java/javame/index.htm. Juni 2012

[6] Anonim. Pengertian Internet, http://id.wikipedia.org/wiki/Internet. Juli 2012

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

NAMA : DWI ARIANTO
NIM : 06.12.618
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer dan Informatika
MASA BIMBINGAN : 4 Juli 2012 s/d 4 Januari 2013
JUDUL : **RANCANG BANGUN APLIKASI EMAIL CLIENT MENGGUNAKAN JAVA 2 PLATFORM MICRO EDITION (J2ME) PADA MOBILE PHONE.**

Dipertahankan dihadapan Majelis Penguji Skripsi Jenjang Strata Satu (S-1) pada :

Hari : Rabu
Tanggal : 08 Agustus 2012
Dengan Nilai : 80,5 (A)

PANITIA UJIAN SKRIPSI

Ketua Majelis Penguji,

Ir. Yusuf Ismail Nakhoda, MT
NIP.Y.1018800189

Sekretaris Majelis Penguji,

Dr. Eng. Aryuanto S, ST, MT
NIP.P.1030800417

ANGGOTA PENGUJI

Dosen Penguji I

Ir. Eko Nurcahyo
NIP.Y.1028700172

Dosen Penguji II

Yuli Wahyuni, ST, MT
NIP.P.1031200456

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan ujian skripsi jenjang Strata Satu (S-1) Jurusan Teknik Elektro Konsentrasi Teknik Komputer dan Informatika, maka perlu adanya perbaikan skripsi untuk mahasiswa :

NAMA : Dwi Arianto
NIM : 06.12.618
JURUSAN : Teknik Elektro S-1
KONSENTRASI : Teknik Komputer dan Informatika
MASA BIMBINGAN : 4 Juli 2012 s/d 4 Januari 2013
JUDUL : **RANCANG BANGUN APLIKASI EMAIL CLIENT MENGGUNAKAN JAVA 2 PLATFORM MICRO EDITION (J2ME) PADA MOBILE PHONE.**

Tanggal	Uraian	Paraf
Penguji I 8-8-2012	Screenshot Koneksi	
Penguji II 8-8-2012	-	

Mengetahui,

Dosen Pembimbing I

(Joseph Dedy Irawan, ST, MT)
NIP.Y.1974041620011002

Dosen Pembimbing II

(Bima Aulia Firmandani, ST)
1121

Disetujui,

Dosen Penguji I

Ir. Eko Nurcahyo
NIP.Y.1028700172

Dosen Penguji II

Yuli Wahyuni, ST, MT
NIP.P.1031200456

1. IMAP 4 Source Code :

```
/*

AngelMail - Simple mail client for J2ME

 */

import java.io.*;
import javax.microedition.io.*;
import java.util.*;
import javax.microedition.lcdui.*;

public class IMAP4 extends InProtocol {

      int commandCounter = 0;
      //name of a selected IMAP4 folder followed with it's UIDVALIDITY
encapsulated within two slashes //
      String sld_mailBox_iudval; //so looks like: INBOX/12313/
      Vector deleted;
      Timer connectionKeeper;
      private class Keeper extends TimerTask {
            public void run() { //sents a command to check and keeps
the connection alive to avoid long reconnecting
                  System.out.println(account.email+": keeping connection
alive...");
                  if (isBusy())
                        return;
                  lock();
                  if (!isConnected()) {
                        _close();
                  }
                  unlock();
            }
      }

      public IMAP4(MailAccount account, InBox box) {
            super(account, box);
            END_OF_MAIL = ")\r\n";
            deleted = new Vector();
      }

      public void addDeleted(MessageHeader header) {
            deleted.addElement(header);
      }

      //Send to the server a command.
      //If resultOnly is true it flushes all response lines to the last
response beginning with the tag without the tag
      private String execute(String command, boolean resultOnly) throws
MyException{
            String tag = "A"+ commandCounter++ +" ";
```

```
            connection.sendCRLF(tag+command);
            if (resultOnly) {
                  String reply = connection.getLine();
                  while (!reply.startsWith(tag))
                        reply = connection.getLine();

                  return reply.substring(tag.length());
            }
            return tag;
      }

      //send to the server the command and returns the result that is
mostly closed in brackets;
      private String execute(String command, String arguments) throws
MyException {
            String tag = "A" + commandCounter++ + " ";
            connection.sendCRLF(tag + command + (arguments == null ? ""
: " " + arguments));
            String result = "";

            String temp = connection.getLine();
            while (!temp.startsWith(tag)) { //multiline response
                  if (temp.indexOf(" " + command + " ") != -1) { //is it
the response for the executed command?
                        int p = temp.indexOf('(');
                        int q = temp.indexOf(')', p + 1);

                        if (p != -1 && q > p) {     //a response in the
form of "* command (result)"
                              result = temp.substring(p + 1, q);

                        } else //a response in the form of "* command
result"
                              result +=
temp.substring(2+command.length()+1); //2 - "* ", 1 - the space after
the command
                  }
                  temp = connection.getLine();
            }

            temp = temp.substring(tag.length());
            if (temp.startsWith("BAD ") || temp.startsWith("NO ")) {
                  throw new
MyException(MyException.PROTOCOL_COMMAND_NOT_EXECUTED);
            }

            return result;
      }

      public boolean isConnected(){
            try {
                  if (connection.isConnected()) {
```

```
                    connection.clearInput();
                    if (execute("NOOP", true).startsWith("OK"))
                        return true;
                }
            } catch (Exception ex) {}
            return false;
        }

        public int countNew() throws MyException {
            return searchMailsMatching("UNSEEN").size();
        }

        //return Vector having UID of mails matching the concrete
criteria
        private Vector searchMailsMatching(String criteria) throws
MyException {
            Vector mails = new Vector();
            execute("UID SEARCH "+criteria, false);

            String line = connection.getLine();
            line = line.substring( line.indexOf("SEARCH")+7 ).trim()+"
";
            int i = 0, j = 0;
            try {
                while(line.length()-1 > i) {
                    j = line.indexOf(" ", i);
                                        ;

                    mails.addElement(line.substring(i, j));
                    i = j+1;
                }
            } catch (Exception ex) {
            } finally {
                return mails;
            }
        }

        private String parseUID(String s) {
            return s.substring(s.lastIndexOf('/')+1);
        }

         private boolean open() throws MyException {
            if(isConnected())
                return true;
            _close(); //we'd better close inactive connections
            try {
                commandCounter = 0;//reset the counter for each
connection
                String reply;

                box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+"
"+account.email);
```

```
                connection.open(account.server + ":" + account.port,
account.SSL);
                if (connection.getLine().length() == 0) {

     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.FAILED));
                     return false;
                }
                reply = execute("LOGIN "+'"'+account.userName+'"'+"
"+'"'+account.password+'"', true);

                if (!reply.startsWith("OK")) {
                     box.report("100:
"+Lang.get(Lang.PL_NOTAUTHORIZED)+" "+reply);
                     return false;
                }


     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.SUCCESS));
                connectionKeeper = new Timer();
                connectionKeeper.scheduleAtFixedRate(new Keeper(),
Settings.noopIMAPPeriod, Settings.noopIMAPPeriod);
                return true;

           } catch(MyException e) {

     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.FAILED));
                throw e;
           } catch (Exception e) {

     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.FAILED));
                throw new MyException(MyException.COM_UNKNOWN, "100:
"+e);
           }
     }

     private boolean selectMailBox(String box) throws MyException {
           String tag = execute("SELECT "+box, false);
           sld_mailBox_iudval = null; //SELECT automatically deselects
currently selected mailbox
           String reply, uidvalidity = "";
           while (  !(reply = connection.getLine()).startsWith(tag) )
{
                if (reply.indexOf("[UIDVALIDITY") != -1)    {
                     uidvalidity = reply.substring(
reply.indexOf("[UIDVALIDITY")+13).trim();
                     uidvalidity = uidvalidity.substring(0,
uidvalidity.indexOf(']')).trim();
                }
```

```
            }
            if (reply.startsWith(tag+"OK")) {
                  sld_mailBox_iudval = box + "/" + uidvalidity + "/";
                  return true;
            }
            return false;
      }

       private synchronized void _close() {
            if (!connection.isConnected())
                  return;

      box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email);
            if (connectionKeeper != null)
                  connectionKeeper.cancel();
            connectionKeeper = null;
            forcedDisc = false;
            sld_mailBox_iudval = null;
            try {
                  execute("CLOSE", true);
                  execute("LOGOUT", true);
            } catch (MyException e) {

      box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email+":
"+e.getDetails() );
            }
            try {
                  connection.close();
            } catch(Exception e) {

      box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email+":
"+e);
            }

      box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email+Lang.
get(Lang.SUCCESS));
      }


      public void run() {

            //polling is an extra thread, is not counted by inThread()
or inThreadGlobal()
            if (runMode == InProtocol.POLL) {
                  Vector newMails = null;
                  String ID = null;

                  lock();
                  try {
                        connection.unQuit();
                        connection.clearInput();
                        if (open())
```

```
                              newMails = searchMailsMatching("UNSEEN");

                         for (int i = newMails.size(); i > 0; --i) {
                              ID = (String)newMails.elementAt(i-1);

     if(!box.wasOnceDownloaded(account.email+"@"+ID)) {

                                   if (Settings.pollDownloadsMails)
                                        getNewMails();
                                   if (Settings.pollPlaysSound)
                                        new AudioAlert();

                                   break;
                              }
                         }

                    } catch (MyException ex) {
                         //do nothing, be silent bitch
                    } catch (Exception ex) {}
                    unlock();
                    return;
               }

               //if its a forced disconnect, then we will not wait for
someone by calling lock()
               if (runMode == InProtocol.CLOSE && forcedDisc) {

     box.setProgress(Lang.get(Lang.ALRT_PL_CLOSING)+account.email,1,0)
;
                    _close();

     box.setProgress(Lang.get(Lang.ALRT_PL_CLOSING)+account.email+Lang
.get(Lang.SUCCESS),1,1);
                    forcedDisc = false;
                    decThreadGlobal();
                    box.repaint();
                    return;
               }

               try {
                    //we use lock() instead of making run() synchronized
in order to allow forced disconnecting from servers
                    //even some jobs are still running
                    lock();
                    inThread();

     box.report(Lang.get(Lang.ALRT_INITIATING)+Lang.get(Lang.ALRT_WAIT
));
                    connection.unQuit();
                    connection.clearInput();
                    long startTime, wholeTime;
                    String waitTime;
```

```
                int actual = 0, max = 0;
                String line, sld_box;
                switch(runMode) {
                     case InProtocol.CLOSE:
                          _close();
                          break;
                     case InProtocol.GET_NEW_MAILS:
                          startTime = System.currentTimeMillis();
                          if (!open()) { //in case of server->inbox
sync we need to notify about this error
                               //otherwise the synchronization will
think that no mails are on the server
                               throw new
MyException(MyException.PROTOCOL_CANNOT_CONNECT);
                          }
                          if (box.mailsOnServers != -1) //if sync is
running
                               mailsOnServer.clear();//we need to
recreate a new mailsOnServer lists

                          //if its server->inbox sync is called then
we want all mails
                          //otherwise we just want to check new mails
                          String criterium = box.mailsOnServers == -
1? "UNSEEN" : "ALL";

     box.report(Lang.get(Lang.ALRT_INPL_CHECK_MAILS)+account.email);

                          MessageHeader header;
                          String mailBoxes = account.IMAP_PRIMARYBOX;
                          if (!mailBoxes.endsWith(","))
                               mailBoxes += ",";


                          while (mailBoxes.length() != 0) {
                               //choose next mailbox
                               sld_box = mailBoxes.substring(0,
mailBoxes.indexOf(','));
                               //define what's the next mailbox
                               mailBoxes =
mailBoxes.substring(sld_box.length()+1);
                               //if selecting a mailbox is
unsuccessful skip to the next one
                               if (!selectMailBox(sld_box))  {

     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+":
"+sld_box+Lang.get(Lang.FAILED));
                                    continue;
                               }
                               Vector newMails = new Vector();
                               Vector fetchTags = new Vector();
                               String fetchTag;
```

```
                                    //let's find UIDs of mails

                                    Vector tmp =
searchMailsMatching(criterium);
                                    for (int i = 0; i < tmp.size(); ++i) {


                                         header = new MessageHeader();
                                         header.messageID =
sld_mailBox_iudval+tmp.elementAt(i);
                                         newMails.addElement(header);
//mark it to potentially new mails
                                    }
                                    int n = 0;
                                    box.updateMax(newMails.size());
                                    //for all potentially new mails, the
newest first
                                    for (actual = newMails.size()-1;
actual >=0; --actual) {
                                         header =
(MessageHeader)newMails.elementAt(actual);

                                         //let's remember that this mail
is stored on the server

     mailsOnServer.put(header.messageID, String.valueOf(actual));
                                         box.newMailOnServer();
//increase synchronization counter
                                         if
(!box.wasOnceDownloaded(account.email+"@"+header.messageID)) { //check
if that mail wasnt already downloaded



                                              fetchTag = execute("UID
FETCH " +parseUID(header.messageID)+" (RFC822.SIZE)", false);

                                              if (!Settings.downWholeMail
|| Settings.safeMode)
                                                   execute("UID FETCH
"+parseUID(header.messageID)+" (RFC822.HEADER)", false);
                                              else {

                                                   execute("UID FETCH
"+parseUID(header.messageID)+" (RFC822)", false);
                                              }

     fetchTags.addElement(fetchTag);
                                              ++n;
```

```
                                        if
(Settings.maxMailsRetrieve > 0 && n >= Settings.maxMailsRetrieve )
//limit excessed
                                            break;
                                    } else { //if the mail was
allready downloaded, remove it from

     newMails.removeElementAt(actual); //remove it from potentially
new mails list
                                        box.updateActual();
                                    }

                                }
                                int j = fetchTags.size()-1;
                                n = newMails.size() - n; //we will
parse only mails whose size and header were required
                                for (actual = newMails.size()-1;
actual >= n; --actual) { //now lets parse new mails' headers
                                    header =
(MessageHeader)newMails.elementAt(actual);
                                    fetchTag =
(String)fetchTags.elementAt(j--);
                                    do {
                                        //skip useless lines of
previous iteration or of fetch response
                                        //until we get tagged BAD
or NO response or good UID response
                                        line =
connection.getLine();
                                    } while (!
(line.startsWith(fetchTag) || line.indexOf("UID
"+parseUID(header.messageID)) != -1));
                                    if ( line.startsWith(fetchTag) )
{ //bad response


     box.report("*"+account.email+": "+line);
                                        continue;
                                    }


                                    int i =
line.indexOf("RFC822.SIZE")+12;
                                    line = line.substring(i);
                                    for (i = 0; i < line.length();
++i)
                                        if ( !('0' <=
line.charAt(i) && line.charAt(i) <= '9') )
                                            break;
                                    header.size = Integer.parseInt(
line.substring(0, i ) );
```

```
                                        parseHeaders(header);

                                        if (Settings.downWholeMail &&
!Settings.safeMode) {

     box.report(Lang.get(Lang.ALRT_INPL_DOWN_MAIL)+header.subject);

                                             try {
                                                  parseBody(header);

     box.report(Lang.get(Lang.ALRT_INPL_DOWN_MAIL)+header.subject+Lang
.get(Lang.SUCCESS));
                                             } catch (MyException me) {



     box.report(Lang.get(Lang.ALRT_INPL_DOWN_MAIL)+header.subject+"
"+Lang.get(Lang.FAILED)+" "+me.getDetails());

                                             }

                                        }
                                        try {

     box.mailDB.saveHeader(header);
                                             //cache the mail so next
time we can quickly recognize it as already downloaded

     box.addToOnceDownloaded(header);
                                        } catch (MyException exp) {
                                             clear(header); //delete
partially downloaded bodies

     box.report(Lang.get(Lang.ALRT_SAVING)+header.subject+"
"+Lang.get(Lang.FAILED)+" "+exp.getDetails());
                                             if (box.mailsOnServers != -
1) //something's gone wrong, now we have to stop sync
                                                  throw exp;

                                        }
                                        //also mark this mail as checked
                                        box.addToMsgIDs(header);
                                        box.addToStorage(header);
//store the mail
                                        if (header.readStatus ==
MessageHeader.NOT_READ)
                                             box.changeUnreadMails(1);
                                        box.updateActual();


                                   }
                              }
```

```
                              wholeTime = System.currentTimeMillis()-
startTime;
                              waitTime = wholeTime  > 1000?
wholeTime/1000+"sec": wholeTime+"msec";

     box.report(Lang.get(Lang.ALRT_INPL_CHECK_MAILS)+account.email+
""+Lang.get(Lang.IN)+waitTime);

                              break;

                         case InProtocol.REDOWNLOAD_BODY:
                         case InProtocol.RETRIEVE_BODY:

                              startTime = System.currentTimeMillis();
                              if (!open()) {
                                   synchronized (actHeader) {
                                        actHeader.notify();
                                        actHeader = null;
                                   }
                                   break;
                              }

     box.setProgress(Lang.get(runMode==RETRIEVE_BODY?
Lang.ALRT_INPL_DOWN_MAIL:Lang.ALRT_INPL_REDOWN_MAIL)+actHeader.subject
,actHeader.size,0);


                              sld_box = actHeader.messageID.substring(0,
actHeader.messageID.lastIndexOf('/')+1 );
                              //if the actually selected mailbox differs
from header's mailBox
                              if (sld_mailBox_iudval == null ||
!sld_mailBox_iudval.equals(sld_box)) {
                                   //select header's mailBox
                                   if
(!selectMailBox(sld_box.substring(0, sld_box.indexOf('/') )) )
                                        throw new
MyException(MyException.PROTOCOL_CANNOT_RETRIEVE_BODY,
Lang.get(Lang.ALRT_INPL_IMAP_CANNOT_SELECT_MAILBOX)+sld_box.substring(
sld_box.indexOf(0, '/')) );


                                   if
(!sld_mailBox_iudval.equals(sld_box))
                                        throw new
MyException(MyException.PROTOCOL_CANNOT_RETRIEVE_BODY,
Lang.get(Lang.ALRT_INPL_IMAP_UIDVALIDITY_DIFFERS) );


                              }
                              String fetchTag;
```

```
                              if (runMode == RETRIEVE_BODY || (runMode ==
REDOWNLOAD_BODY && reDownloadMode == -1))
                                   fetchTag = execute("UID FETCH
"+parseUID(actHeader.messageID)+" (RFC822)", false);
                              else
                                   fetchTag = execute("UID FETCH
"+parseUID(actHeader.messageID)+" BODY"+"["+(reDownloadMode+1)+"]",
false);

                              do { //skip useless lines fetch response
                                   line = connection.getLine();

                              } while (! (line.startsWith(fetchTag) ||
line.indexOf("UID "+parseUID(actHeader.messageID)) != -1));
                              if ( line.startsWith(fetchTag) )  {

                                   throw new
MyException(MyException.PROTOCOL_CANNOT_RETRIEVE_BODY, "200:
"+Lang.get(Lang.ALRT_INPL_NO_LONGER_ON_SERVER));


                              }

                              parseBody(actHeader);

                              box.mailDB.saveHeader(actHeader); //update
new data into DBase
                              if (Settings.safeMode)
                                   box.lastSafeMail = actHeader;

                              wholeTime = System.currentTimeMillis() -
startTime;
                              waitTime = wholeTime >1000?
wholeTime/1000+"sec": wholeTime+"msec";

     box.setProgress("*"+Lang.get(runMode==RETRIEVE_BODY?
Lang.ALRT_INPL_DOWN_MAIL:Lang.ALRT_INPL_REDOWN_MAIL)+actHeader.subject
+" "

     +Lang.get(Lang.IN)+waitTime,actHeader.size,actHeader.size);




                              synchronized (actHeader) { //everything is
ok now
                                   actHeader.notify();
                                   actHeader = null;
                              }
```

```
                    break;

                case InProtocol.REMOVE_MAILS:

                    startTime = System.currentTimeMillis();
                    if (deleted.isEmpty() || !open())
                        break;

    box.report(Lang.get(Lang.ALRT_INPL_DEL_MAILS)+account.email);
                    box.updateMax(deleted.size());
                    //let's sort the marked mails by their
mailboxes
                    //so we don't have to reselect mailboxes
for every mail having the same mailbox
                    Functions.sort(deleted,
Functions.SRT_ORDER_INC, Functions.SRT_HDR_MSGID);

                    String msgID;
                    max = deleted.size();
                    actual = 0;
                    int i = 0, j = 0;
                    for (actual = 0; actual < max; ++actual) {
                        msgID =
((MessageHeader)deleted.elementAt(actual)).messageID;

                        sld_box = msgID.substring(0,
msgID.lastIndexOf('/')+1 ); //this mail's mail box

                        j = sld_box.indexOf('/');
                        //if we're gonna select another
mailbox, expunge the currently selected mailbox
                        if (i != 0 && (i != j ||
!sld_mailBox_iudval.regionMatches(false, 0, sld_box, 0, j)) )
                            execute("EXPUNGE", true);

                        //if no mailbox was set or the
actually selected mailbox differs from header's mailBox
                        if (i != j ||
!sld_mailBox_iudval.regionMatches(false, 0, sld_box, 0, j) ) {


                            i = 0;
                            //select header's mailBox
                            if
(!selectMailBox(sld_box.substring(0, j)) ) {
                                box.report(
"*"+Lang.get(Lang.ALRT_INPL_IMAP_CANNOT_SELECT_MAILBOX)+sld_box.substr
ing(sld_box.indexOf(0, '/')) );


                                //dont consider this as
deleted from the server
```

```
                                        //and don't remove it from
the onceDownloaded cache to prevent redownloading it again

      deleted.removeElementAt(actual);


                                        continue;
                                    }
                                    i =
sld_mailBox_iudval.indexOf('/');
                                }

                                //if the mailbox has changed its
UIDVALIDITY from the previous session
                                if
(!sld_mailBox_iudval.equals(sld_box)) {
                                    box.report(
"*"+Lang.get(Lang.ALRT_INPL_IMAP_UIDVALIDITY_DIFFERS) );


                                    deleted.removeElementAt(actual);

                                }
                                else
                                    execute("UID STORE "+
parseUID(msgID) +" +FLAGS (\\Deleted)", false);

                            }

                            if (actual != 0)
                                execute("EXPUNGE", true);

                            //for all successfuly deleted mails
                            for (actual = deleted.size(); actual > 0; -
-actual) {
                                msgID =
((MessageHeader)deleted.elementAt(actual-1)).messageID;
                                mailsOnServer.remove(msgID);

      box.removeOnceDownloaded(account.email+"@"+msgID);
                                box.updateActual();

                            }
                            deleted.removeAllElements();

                            wholeTime = System.currentTimeMillis() -
startTime;
                            waitTime = wholeTime > 1000?
wholeTime/1000+"sec": wholeTime+"msec";

      box.report("*"+Lang.get(Lang.ALRT_INPL_DEL_MAILS)+account.email+L
```

```
ang.get(Lang.IN)+waitTime);

                              break;

                    }

               } catch (MyException ex) {
                    if (ex.errorCode == MyException.COM_HALTED)
                         connection.unQuit();
                    resolveExceptions(ex.getDetails()+"/ "+account.email);
               } catch (Error e) {
                    resolveExceptions("100: "+e+"/ "+account.email);

               } catch (Exception ex) {
                    resolveExceptions("100: "+ex+"/ "+account.email);

               }

               decThread();
               decThreadGlobal();
               unlock();
               synchronized (box) {
                    //if im the last headerRetrieving thread, i should
resort the box
                    if (box.needResort && !InProtocol.isBusyGlobal()) {
                         inThreadGlobal(); //this will make thebox trigger
the progress bar
                         box.resort();
                         box.needResort = false;
                         box.setCurFirstUnread();
                         decThreadGlobal();
                    }
                    //if its servers sync notify the box to completed the
sync process
                    if (box.mailsOnServers != -1)
                         box.serversSync();
               }
               box.repaint();
          }

}
```

2. **POP 3 Source Code :**

```
/*

AngelMail - Simple mail client for J2ME
```

```
 */


import java.io.*;
import javax.microedition.io.*;
import java.util.*;
import javax.microedition.lcdui.*;

public class POP3 extends InProtocol {

      Hashtable deleted;
      Timer connectionKeeper;
      private class Keeper extends TimerTask {
            public void run() { //sents a command to check and keeps
the connection alive to avoid long reconnecting
                  System.out.println(account.email+": keeping connection
alive...");
                  if (isBusy())
                        return;
                  lock();
                  if (!isConnected()) {
                        _close();
                  }
                  unlock();
            }
      }

      public POP3(MailAccount account, InBox box) {
            super(account, box);
            END_OF_MAIL = ".\r\n";
            deleted = new Hashtable();
      }

      //add a mail to a queue of mails that are going to be deleted
from the server
      public void addDeleted(MessageHeader header) {
            deleted.put(header.messageID, header);
      }
      public boolean isConnected(){
            try {
                  if (connection.isConnected()) {
                        connection.clearInput();
                        connection.sendCRLF("NOOP");
                        if (connection.getLine().startsWith("+OK"))
                              return true;
                  }
            } catch (Exception ex) {}
            return false;
      }

      public int countNew() throws MyException {
```

```
            connection.sendCRLF("STAT");
            String reply = connection.getLine();
            if (!reply.startsWith("+OK")) {
                  return 0;
            }
            return Integer.parseInt( reply.substring(4,
reply.lastIndexOf(' ')).trim() );
      }

      private int getSize(int msgNum) throws MyException {
            connection.sendCRLF("LIST " + msgNum);
            String reply = connection.getLine();
            if( !reply.startsWith("+OK") ) {
                  return 0;
            }
            return Integer.parseInt(
reply.substring(reply.lastIndexOf(' ') + 1).trim() );
      }

      private String getMsgID(int index) throws MyException {
            connection.sendCRLF("UIDL "+index);
            String reply = connection.getLine();
            if (reply.startsWith("+OK"))
                  return reply.substring( reply.lastIndexOf(' ') +
1).trim();
            else return "";
      }

      private String getMsgNum(MessageHeader header) throws
MyException{
            String reply, ID, num;
            String index = (String)
mailsOnServer.get(header.messageID);
            if (index != null) {
                  connection.sendCRLF("UIDL "+index); //lets try if the
mail is from actual session
                  reply = connection.getLine();
                  if (reply.startsWith("+OK")) {
                        ID = reply.substring( reply.lastIndexOf(' ')+1
).trim();
                        if (ID.equals(header.messageID))
                              return index;
                  }
            }
            //we have to test and update info about all mails :(
            //note: if a message is marked as deleted or as read by
pop3 , it will never be listed by UIDL again and we can get it
anymore?
            connection.sendCRLF("UIDL");
            connection.getLine();
            boolean found = false;
```

```
            while( !(reply = connection.getLine()).startsWith(".") ) {

                  ID = reply.substring( reply.indexOf(' ')+1 ).trim();
                  num = reply.substring(0, reply.indexOf(' '));


                  if (mailsOnServer.containsKey(ID)) //let's remove old
information
                        mailsOnServer.remove(ID);
                  mailsOnServer.put(ID, num);

                  if (ID.equals(header.messageID))
                        found = true;
            }
            return found? (String)mailsOnServer.get(header.messageID):
"0";
      }

      private boolean open() throws MyException {
            if(isConnected())
                  return true;
            _close(); //we'd better close inactive connections
            try {
                  box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+"
"+account.email);
                  connection.open(account.server + ":" + account.port,
account.SSL);
                  if (!connection.getLine().startsWith("+OK")) {

      box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.FAILED));
                        return false;
                  }
                  connection.sendCRLF("USER "+account.userName);

                  if (!connection.getLine().startsWith("+OK")) {
                        connection.unGetLine();
                        box.report("100:
"+Lang.get(Lang.PL_NOTAUTHORIZED)+account.userName+"/
"+connection.getLine());
                        return false;
                  }
                  connection.sendCRLF("PASS "+account.password);

                  if (!connection.getLine().startsWith("+OK")) {
                        box.report("100:
"+Lang.get(Lang.PL_NOTAUTHORIZED)+account.userName);
                        return false;
                  }

      box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.SUCCESS));
```

```
                connectionKeeper = new Timer();
                connectionKeeper.scheduleAtFixedRate(new Keeper(),
Settings.noopPeriod, Settings.noopPeriod);
                return true;

           } catch(MyException e) {

     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.FAILED));
                throw e;
           } catch (Exception e) {

     box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+account.email+Lang.g
et(Lang.FAILED));
                throw new MyException(MyException.COM_UNKNOWN, "100:
"+e);
           }
      }

       private synchronized void _close() {
           if (!connection.isConnected())
                return;
           //+ because we don't want to confuse the user with this
when he's viewing the folder

     box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email);
           if (connectionKeeper != null)
                connectionKeeper.cancel();
           connectionKeeper = null;
           forcedDisc = false;
           try {
                connection.sendCRLF( "QUIT");
           } catch (MyException e) {

     box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email+":
"+e.getDetails() );
           }
           try {
                connection.close();
           } catch(Exception e) {

     box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email+":
"+e);
           }

     box.report("+"+Lang.get(Lang.ALRT_PL_CLOSING)+account.email+Lang.
get(Lang.SUCCESS));
      }

      public void run() {
```

```
          //polling is an extra thread, is not counted by inThread()
or inThreadGlobal()
          //it can be stop only by forced disconnecting from the
server a pressing "Polling" in the menu
          if (runMode == InProtocol.POLL) {
               int count;
               String ID = null;

               lock();
               try {
                    connection.unQuit();
                    connection.clearInput();
                    if (open())
                         count = countNew();
                    else count = 0;
                    for (; count > 0; --count) {
                         ID = getMsgID(count);
                         if(ID.length() > 0 &&
!box.wasOnceDownloaded(account.email+"@"+ID)) {

                              if (Settings.pollDownloadsMails)
                                   getNewMails();
                              if (Settings.pollPlaysSound)
                                   new AudioAlert();

                              break;
                         }
                    }
               } catch (MyException ex) {
                    //do nothing, be silent bitch
               } catch (Exception ex) {}
               unlock();

               return;
          }

          //if its a forced disconnect, then we will not wait for
someone by calling lock()
          if (runMode == InProtocol.CLOSE && forcedDisc) {

     box.setProgress(Lang.get(Lang.ALRT_PL_CLOSING)+account.email,1,0)
;
               _close();

     box.setProgress(Lang.get(Lang.ALRT_PL_CLOSING)+account.email+Lang
.get(Lang.SUCCESS),1,1);
               forcedDisc = false;
               decThreadGlobal();
               box.repaint();
               return;
          }
```

```
          try {
               //we use lock() instead of making run() synchronized
in order to allow forced disconnecting from servers
               //even some job are still running
               lock();
               inThread();

     box.report(Lang.get(Lang.ALRT_INITIATING)+Lang.get(Lang.ALRT_WAIT
));
               connection.unQuit();
               connection.clearInput();
               long startTime, wholeTime;
               String waitTime;
               int actual = 0, max = 0;
               switch(runMode) {
                    case InProtocol.CLOSE:
                         _close();
                         break;
                    case InProtocol.GET_NEW_MAILS:
                         startTime = System.currentTimeMillis();
                         if (!open()) { //in case of server->inbox
sync we need to notify about this error
                              //otherwise the synchronization will
think that no mails are on the server
                              throw new
MyException(MyException.PROTOCOL_CANNOT_CONNECT);
                         }
                         if (box.mailsOnServers != -1) //if sync is
running
                              mailsOnServer.clear();//we need to
recreate a new mailsOnServer lists

                         max = countNew();

     box.report(Lang.get(Lang.ALRT_INPL_CHECK_MAILS)+account.email);
                         box.updateMax(max);
                         while( actual < max ) { //lets send
commands to get msgIDs and size for the mails
                              connection.sendCRLF("UIDL "+
(actual+1) +"\r\n" +"LIST "+ (actual+1));
                              ++actual;
                         }
                         String IDLine, SizeLine;
                         MessageHeader header;
                         Vector newMails = new Vector(actual);
                         while (actual > 0) { //lets parse MsgIDs
and size from input stream
                              IDLine = connection.getLine();
                              SizeLine = connection.getLine();
                              //if we could get its MsgID and size,
put it to newMails - mark it as a potentially new mail
```

```
                                    if (IDLine.startsWith("+OK") &&
SizeLine.startsWith("+OK")) {
                                        header = new MessageHeader();
                                        header.messageID =
IDLine.substring( IDLine.lastIndexOf(' ') + 1).trim();
                                        header.size = Integer.parseInt(
SizeLine.substring(SizeLine.lastIndexOf(' ') + 1).trim() );
                                        newMails.addElement(header);
                                    }
                                    --actual;
                                }
                                int n = 0;
                                for (actual = newMails.size()-1; actual >=
0; --actual) { //for all potentially new mails in the list, the newest
first
                                    header =
(MessageHeader)newMails.elementAt(actual);
                                    mailsOnServer.put(header.messageID,
String.valueOf(actual+1));
                                    box.newMailOnServer();//increase
synchronization counter

                                    if
(!box.wasOnceDownloaded(account.email+"@"+header.messageID)) { //check
if that mail wasnt already downloaded



                                        if (!Settings.downWholeMail ||
Settings.safeMode)
                                            connection.sendCRLF("TOP
"+(actual+1)+" 0");
                                        else {

                                            //we send
connection.sendCRLF("TOP "+n+" 1000000");
                                            //instead of
connection.sendCRLF("RETR "+n); here because RETR returns data in a
messed order
                                            //not the same as we
wanted!!!
                                            //ei: RETR 1,RETR 2, RETR3
are returned as result_of_RETR1 half_of_RETR2, result_RETR3 and then
rest of RETR2
                                            //very annoying thing. I
spent 3 days with this (Tung).
                                            connection.sendCRLF("TOP
"+(actual+1)+" 1000000");
                                        }
                                        ++n;
```

```
                                        if (Settings.maxMailsRetrieve >
0 && n >= Settings.maxMailsRetrieve ) //limit excessed

                                              break;


                                  } else { //if the mail was already
downloaded, remove it from the potentially new mails list

      newMails.removeElementAt(actual);
                                        box.updateActual();
                                  }

                              }
                              n = newMails.size() - n;
                              for (actual = newMails.size()-1; actual >=
n; --actual) { //now lets parse new mails' headers
                                  header =
(MessageHeader)newMails.elementAt(actual);
                                  String line;
                                  //skip useless lines of previous
iteration
                                  do {
                                        line = connection.getLine();

                                  } while (!line.startsWith("+OK") &&
!line.startsWith("-ERR"));
                                  if ( line.startsWith("-ERR") )  {

                                        box.report("*"+account.email+":
"+line);
                                        continue;
                                  }

                                  parseHeaders(header);

                                  if (Settings.downWholeMail &&
!Settings.safeMode) {

      box.report(Lang.get(Lang.ALRT_INPL_DOWN_MAIL)+header.subject);

                                        try {
                                              parseBody(header);

      box.report(Lang.get(Lang.ALRT_INPL_DOWN_MAIL)+header.subject+Lang
.get(Lang.SUCCESS));
                                        } catch (MyException me) {



      box.report(Lang.get(Lang.ALRT_INPL_DOWN_MAIL)+header.subject+"
"+Lang.get(Lang.FAILED)+" "+me.getDetails());
```

```
                                        }

                                    }
                                    try {
                                        box.mailDB.saveHeader(header);
                                        //cache the mail so next time we
can quickly recognize it as already downloaded
                                        box.addToOnceDownloaded(header);
                                    } catch (MyException exp) {
                                        clear(header); //delete
partially downloaded bodies

     box.report(Lang.get(Lang.ALRT_SAVING)+header.subject+"
"+Lang.get(Lang.FAILED)+" "+exp.getDetails());

                                        if (box.mailsOnServers != -1)
//something's gone wrong, now we have to stop sync
                                            throw exp;

                                    }

                                    box.addToMsgIDs(header);
                                    box.addToStorage(header); //store the
mail to box's storage vector
                                    if (header.readStatus ==
MessageHeader.NOT_READ)
                                        box.changeUnreadMails(1);
                                    box.updateActual();

                                }

                                wholeTime = System.currentTimeMillis()-
startTime;
                                waitTime = wholeTime  > 1000?
wholeTime/1000+"sec": wholeTime+"msec";

     box.report("*"+Lang.get(Lang.ALRT_INPL_CHECK_MAILS)+account.email
+ ""+Lang.get(Lang.IN)+waitTime);
                                break;

                            case InProtocol.REDOWNLOAD_BODY:
                            case InProtocol.RETRIEVE_BODY:

                                startTime = System.currentTimeMillis();
                                if (!open()) {
                                    synchronized (actHeader) {
                                        actHeader.notify();
                                        actHeader = null;
                                    }
                                    break;
                                }
```

```
     box.setProgress(Lang.get(runMode==RETRIEVE_BODY?
Lang.ALRT_INPL_DOWN_MAIL:Lang.ALRT_INPL_REDOWN_MAIL)+actHeader.subject
,actHeader.size,0);


                              connection.sendCRLF("RETR
"+getMsgNum(actHeader));
                              if (connection.getLine().startsWith("-
ERR"))
                                   throw new
MyException(MyException.PROTOCOL_CANNOT_RETRIEVE_BODY, "200:
"+Lang.get(Lang.ALRT_INPL_NO_LONGER_ON_SERVER));


                              parseBody(actHeader);

                              box.mailDB.saveHeader(actHeader); //update
new data into DBase
                              if (Settings.safeMode)
                                   box.lastSafeMail = actHeader;
                              wholeTime = System.currentTimeMillis() -
startTime;
                              waitTime = wholeTime >1000?
wholeTime/1000+"sec": wholeTime+"msec";

     box.setProgress("*"+Lang.get(runMode==RETRIEVE_BODY?
Lang.ALRT_INPL_DOWN_MAIL:Lang.ALRT_INPL_REDOWN_MAIL)+actHeader.subject
+" "

     +Lang.get(Lang.IN)+waitTime,actHeader.size,actHeader.size);




                              synchronized (actHeader) { //everything is
ok now
                                   actHeader.notify();
                                   actHeader = null;
                              }


                              break;

                         case InProtocol.REMOVE_MAILS:

                              startTime = System.currentTimeMillis();
                              if (deleted.isEmpty() || !open())
                                   break;
                              max = deleted.size();
                              actual = 0;
```

```
box.report(Lang.get(Lang.ALRT_INPL_DEL_MAILS)+account.email);
                        box.updateMax(max);
                        connection.sendCRLF("UIDL");
                        connection.getLine();
                        String reply, ID, num;
                        //browse the whole UIDL list and try to
delete all marked mails
                        while( !(reply =
connection.getLine()).startsWith(".")) {
                              ID = reply.substring( reply.indexOf('
')+1 ).trim(); //get its ID on the server
                              if (deleted.containsKey(ID)) { //was
is marked as deleted?
                                    num = reply.substring( 0,
reply.indexOf(' ')).trim(); //get its number offset on the server
                                    connection.sendCRLF("DELE
"+num);
                                    mailsOnServer.remove(ID);

box.removeOnceDownloaded(account.email+"@"+ID);
                                    box.updateActual();

                                    actual++;
                              }
                        }
                        while (actual-- > 0)
                              connection.getLine(); //get server's
response line.
                              //We don't check the responses here
just because of laziness :)
                        _close();
                        deleted.clear(); //all mails marked as
deleted should be deleted by now

                        wholeTime = System.currentTimeMillis() -
startTime;
                        waitTime = wholeTime > 1000?
wholeTime/1000+"sec": wholeTime+"msec";

box.report("*"+Lang.get(Lang.ALRT_INPL_DEL_MAILS)+account.email+L
ang.get(Lang.IN)+waitTime);
                        break;

                  }

            } catch (MyException ex) {
                  if (ex.errorCode == MyException.COM_HALTED)
                        connection.unQuit();
                  resolveExceptions(ex.getDetails()+"/ "+account.email);
            } catch (Error e) {
```

```
                resolveExceptions("100: "+e+"/ "+account.email);

            } catch (Exception ex) {
                resolveExceptions("100: "+ex+"/ "+account.email);

            }

            decThread();
            decThreadGlobal();
            unlock();
            synchronized (box) {
                //if im the last headerRetrieving thread, i should
resort the box
                if (box.needResort && !InProtocol.isBusyGlobal()) {
                    inThreadGlobal(); //this will make thebox trigger
the progress bar
                    box.resort();
                    box.needResort = false;
                    box.setCurFirstUnread();
                    decThreadGlobal();
                }
                //if its servers sync notify the box to completed the
sync process
                if (box.mailsOnServers != -1)
                    box.serversSync();
            }
            box.repaint();
        }

} //end of POP3 class
```

3. **SMTP Source Code :**

```
/*

AngelMail - Simple mail client for J2ME

 */


import java.io.*;
import javax.microedition.io.*;
import java.util.*;

public class SMTP implements Runnable {

    public static final byte CLOSE = 1;
```

```
	public static final byte SEND = 2;
	byte runMode;

	AngelMail angelMail;
	TheBox box;
	boolean busy;
	MessageHeader singleMail; //if we just want to send a single mail
	BasicConnection connection;
	boolean locked; //for thread synchronizing
	boolean forcedDisc; //if it should disconnect from the server
unconditionally

	public SMTP(TheBox box, AngelMail angelMail) {
		this.box = box;
		this.angelMail = angelMail;
		connection = new SocketConnection();
	}

	protected synchronized void lock() {
		try {
			while(locked) {
				wait(100);
			}
		} catch(InterruptedException e) {
		}
		locked=true;
	}
	protected void unlock() {
		locked=false;
	}

	public void stop() {
		if (isBusy())
			connection.quit();
	}

	public boolean isConnected(){
		try {
			if (connection.isConnected()) {
			    connection.sendCRLF("NOOP");
			    if (connection.getLine().compareTo("") != 0)
					return true;
			}
		} catch (Exception ex) {
			return false;
		}
		return false;
	}

	private boolean open() throws MyException {
		if (isConnected())
			return true;
```

```
            _close();
            try {

      box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+Settings.smtpServer)
;
                  connection.open(Settings.smtpServer + ":" +
Settings.smtpPort, Settings.smtpSSL);
                  box.report("Server: "+connection.getLine());
                  box.report("Saying HELO");
                  connection.sendCRLF("HELO angelmail.xf.cz");
                  box.report("Server: " +connection.getLine());
                  if (Settings.smtpAuthName.length() != 0) { //todo:
test this and return false/true
                        box.report("Athorizing...");
                        connection.sendCRLF("AUTH PLAIN " +
Decode.toBase64("\000"+ Settings.smtpAuthName+"\000" +
Settings.smtpAuthPass, false) );
                        box.report("Server: "+connection.getLine());
                  }

      box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+Settings.smtpServer+
Lang.get(Lang.SUCCESS));
            } catch(MyException e) {

      box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+Settings.smtpServer+
Lang.get(Lang.FAILED));
                  throw e;
            } catch (Exception e) {

      box.report(Lang.get(Lang.ALRT_PL_CONNECTING)+Settings.smtpServer+
Lang.get(Lang.FAILED));
                  throw new MyException(MyException.COM_UNKNOWN, "100:
"+e.toString());
            }
            return true;
      }

      private synchronized void _close() {
            if (!connection.isConnected())
                  return;
            box.report(
Lang.get(Lang.ALRT_PL_CLOSING)+Settings.smtpServer);
            forcedDisc = false;
            try {
                  connection.sendCRLF( "QUIT");
            } catch (MyException e) {
                  box.report(
Lang.get(Lang.ALRT_PL_CLOSING)+Settings.smtpServer+": "+e.getDetails()
);
            }
            try {
                  connection.close();
```

```
            } catch(Exception e) {
                box.report(
Lang.get(Lang.ALRT_PL_CLOSING)+Settings.smtpServer+": "+e);
            }
            box.report(
Lang.get(Lang.ALRT_PL_CLOSING)+Settings.smtpServer+Lang.get(Lang.SUCCE
SS));
      }

      public boolean isBusy() {
            return busy;
      }

      public synchronized void sendMails() {
            runMode = SEND;
            Thread t = new Thread(this);
            t.start();
            t.setPriority(Thread.MAX_PRIORITY);
      }

      public synchronized void close(boolean forcedDisc) {
            if (this.forcedDisc)
                  return;
            this.forcedDisc = forcedDisc;
            runMode = CLOSE;
            Thread t = new Thread(this);
            t.start();
            t.setPriority(Thread.MAX_PRIORITY);
      }

      public void run() {
            if (runMode == CLOSE && forcedDisc) {
                  busy = true;
                  _close();
                  forcedDisc = false;
                  busy = false;
                  box.repaint();
                  return;
            }
            lock();
            busy = true;
            box.addCommand(box.stop);
            connection.unQuit();
            switch (runMode) {
                  case CLOSE:
                        _close();
                        break;
                  case SEND:
                        Vector rcps = null;
                        String tmpRcp;
```

```
                    short max = (short)box.storage.size();


                    if (singleMail != null)

     box.setProgress(Lang.get(Lang.ALRT_SMTP_SENDING), 1, 0);
                    else

     box.setProgress(Lang.get(Lang.ALRT_SMTP_SENDING), max, 0);
                    MessageHeader message;
                    try {
                         connection.clearInput();
                         open();
                         for (short j = max; j > 0; j--) {

                              if (singleMail != null)
                                   message = singleMail;
                              else
                                   message =
(MessageHeader)box.storage.elementAt(j-1);

     box.report(Lang.get(Lang.ALRT_SMTP_SENDING)+" "+message.subject);
                              connection.sendCRLF("MAIL FROM:
<"+Functions.emailOnly(message.from)+">");
                              connection.getLine();

                              rcps = message.getRcp();

                              for (short i = (short)(rcps.size() -
1); i >= 0; --i) {
                                   tmpRcp =
(String)rcps.elementAt(i);
                                   if (Settings.addToAddressbook)
//add recipients to addressbook
                                        try{

     angelMail.addressBook.saveContact(new
AddressBook.Contact("",tmpRcp,"") );
                                        } catch (MyException ex) {}



                                   connection.sendCRLF("RCPT TO:
<"+tmpRcp+">");
                                   connection.getLine();
                              }

                              connection.sendCRLF("DATA");
                              connection.getLine();
                              connection.sendCRLF("Date: " +
message.getTimeStr() + " " + Functions.getLocalTimeZone());
```

```
                                    connection.sendCRLF("From:
"+message.from);
                                    short i;
                                    i = (short)
message.recipients.indexOf("To:");
                                    if (i != -1) {
                                          tmpRcp =
Functions.encodeRcpNames( message.recipients.substring(i,
message.recipients.indexOf(" *",i+3)) );
                                          connection.sendCRLF(tmpRcp);
                                    }
                                    i = (short)
message.recipients.indexOf("Cc:");
                                    if (i != -1) {

                                          tmpRcp =
Functions.encodeRcpNames( message.recipients.substring(i,
message.recipients.indexOf(" *",i+3)) );
                                          connection.sendCRLF(tmpRcp);
                                    }
                                    i = (short)
message.recipients.indexOf("Bcc:");
                                    if (i != -1) {

                                          tmpRcp =
Functions.encodeRcpNames( message.recipients.substring(i,
message.recipients.indexOf(" *",i+4)) );
                                          connection.sendCRLF(tmpRcp);
                                    }
                                    connection.sendCRLF("Subject:
"+Decode.encodeHeaderField(message.subject));


                                    if (message.messageFormat ==
MessageHeader.FRT_PLAIN) {
                                          connection.sendCRLF("MIME-
Version: 1.0");
                                          connection.sendCRLF("Content-
Type: text/plain; charset=UTF-8");
                                          connection.sendCRLF("Content-
Transfer-Encoding: base64");
                                          connection.sendCRLF("Content-
Disposition: inline");
                                    }

                                    connection.sendCRLF("");
                                    if (message.messageFormat ==
MessageHeader.FRT_PLAIN) {
                                          if(message.getBodyPartCount() ==
1) {
                                                String body =
Functions.toCRLF(message.getBodyPartContent((byte)0) +
```

```
(Settings.signature.length() != 0?  "\r\n--\r\n"+Settings.signature :
"")  );
                                        connection.sendCRLF(
Decode.toBase64(body, false) );
                                    }
                                } else { //se udela potom :)
                                }

                                connection.sendCRLF("\r\n.");
                                if
(!connection.getLine().startsWith("250")) {
                                    message.sendStatus =
message.FAILED_TO_SEND;
                                    try {

      box.mailDB.saveHeader(message);
                                    } catch (MyException ex) {}
                                    connection.unGetLine();

      box.report("*"+Lang.get(Lang.ALRT_SMTP_SENDING)+message.subject+L
ang.get(Lang.FAILED)+": "+connection.getLine());

                                    if (singleMail != null)

                                        break;

                                    continue;
                                }
                                box.report(
Lang.get(Lang.ALRT_SMTP_SENDING)+message.subject+Lang.get(Lang.SUCCESS
) );
                                box.updateActual();
                                message.sendStatus = message.SENT;

                                if (!message.deleted)
                                    ++box.deleted;
                                message.deleted = true;

                                if (!Settings.safeMode)

      angelMail.sentBox.storeMail(message);

                                if (singleMail != null)

                                    break;

                            }
                            Functions.sleep(500);
                            angelMail.mainMenu();

                        } catch (MyException ex) {
```

```
                              //_close() prevents a deadlock in the next
session,
                              //as now when we are in the middle of the
trasnsaction, the server is waiting for a datas from us
                              //but the nexr session we are waiting for a
response from the server
                              _close();
                              box.report(ex.getDetails());

                              if (ex.errorCode == MyException.COM_HALTED)
                                   connection.unQuit();
                         } catch (Exception ex) {
                              _close();
                              box.report("100: "+ex);
                         }
                         box.removeCommand(box.stop);
                         box.deleteNow();
                         break;
               }
               busy = false;
               singleMail = null;
               unlock();
               box.repaint();
          }

}
```

4. Socke Connection :

```
/*

AngelMail - Simple mail client for J2ME

 */

import java.io.*;
import javax.microedition.io.*;

public class SocketConnection extends BasicConnection {


      OutputStream outputStream;
      InputStream inputStream;
      StreamConnection streamConnection;

      protected void write(byte[] data) throws IOException {
            outputStream.write(data);
                  outputStream.flush();
      }
```

```
	protected int read(byte[] buffer, int offset, int length) throws
IOException {
		return inputStream.read(buffer, offset, length);
	}

	//returns how many bytes are ready in the inputStream for further
reading
	protected int available() throws IOException {
		return inputStream.available();
	}

	public void close() throws IOException {
		connected = false;
		pos = 0;
		len = 0;
		quit = false;
		streamConnection.close();
		outputStream.close();
		inputStream.close();
	}

	public void open(String url, boolean ssl) throws IOException {

		if (ssl) {
			streamConnection = (StreamConnection)
Connector.open("ssl://" + url);
			available_bug = true;
		}
		else
			streamConnection = (StreamConnection)
Connector.open("socket://" + url, Connector.READ_WRITE, true);
		inputStream = streamConnection.openInputStream();
		outputStream = streamConnection.openOutputStream();
		connected = true;
		pos = 0;
		len = 0;
		quit = false;
	}

}
```