KAMUS BAHASA INGGRIS – INDONESIA DENGAN MENGGUNAKAN JAVA 2 *MICRO EDITION*

(Skripsi)

Oleh
Dwi Puspita Sari

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2010

ABSTRACT

ENGLISH-INDONESIAN DICTIONARY WITH JAVA 2 MICRO EDITION

by :

Dwi Puspita Sari

The utilization of mobile communication is very popular in society, particularly mobile communication grows very quick so that they produce some kind of mobile. One of a mobile technology that very popular is mobile phone. Nowadays, mobile phones grow quickly and can support some activity. Portable dictionary is the one of application that can used by user. Using this dictionary can help user to search the words in English or Indonesian language. The tool that is used to make the program is J2ME (Java 2 Micro Edition). J2ME is a program language that is used to develop an application that will used in mobile phone. Database management with RMS (Record Management System) as a non-volatile storage in MIDlet. One of the alternative to communicate in other language beside dictionary is mobile dictionary.

Keyword : J2ME, mobile phone, RMS

ABSTRAK

KAMUS BAHASA INGGRIS-INDONESIA DENGAN MENGGUNAKAN JAVA 2 *MICRO EDITION*

Oleh :

Dwi Puspita Sari

Penggunaan alat komunikasi *mobile* saat ini sudah sangat marak di kalangan masyarakat, apalagi dengan didukung dengan perkembangan teknologi yang sangat pesat sehingga menghasilkan beragam jenis *mobile*. Salah satu teknologi *mobile* yang paling populer adalah penggunaan telepon selular. Telepon selular saat ini sudah sangat berkembang dengan pesat dan dapat mendukung berbagai macam aktifitas seperti menjadikannya sebagai alat bantu dalam beberapa hal. Kamus portabel adalah salah satu aplikasi yang dapat ditunjang oleh pemilik telepon selular. Kegunaan kamus ini sendiri adalah dapat mempermudah *user* untuk mencari kata dalam Bahasa Inggris maupun Bahasa Indonesia. Perangkat yang digunakan pada pembuatan program adalah J2ME (Java 2 *Micro Edition*). J2ME merupakan bahasa pemrograman yang digunakan untuk membuat aplikasi yang akan dijalankan pada telepon selular. Pengelolaan *database* dengan menggunakan RMS (*Record Management System*) sebagai media penyimpanan non-*volaltile* dalam MIDlet. Penggunaan *mobile dictionary* ini dapat dijadikan sebagai salah satu alternatif untuk mempermudah dalam pencarian kata dalam bahasa lain.

Kata kunci : J2ME, telepon selular, RMS

KAMUS BAHASA INGGRIS – INDONESIA DENGAN MENGGUNAKAN JAVA 2 *MICRO EDITION*

Oleh
Dwi Puspita Sari

Skripsi

Sebagai Salah Satu Syarat untuk Memperoleh Gelar
SARJANA KOMPUTER

Pada

Program Studi Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2010

PERNYATAAN

Saya yang bertanda tangan dibawah ini, menyatakan bahwa skripsi saya yang berjudul "**KAMUS BAHASA INGGRIS-INDONESIA DENGAN MENGGUNAKAN JAVA 2 *MICRO EDITION* "** ini merupakan hasil karya saya sendiri dan bukan hasil karya orang lain. Semua hasil tulisan yang tertuang dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila dikemudian hari terbukti bahwa skripsi ini berupa salinan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar akademik yang telah saya terima.

Bandar Lampung, November 2010
Yang Membuat Pernyataan

Dwi Puspita Sari

RIWAYAT HIDUP

Penulis merupakan anak kedua dari tiga bersaudara, buah kasih pasangan Sigit Trenggono, SH. dan Yenni Rina. Penulis dilahirkan di Tanjung Karang, 11 Januari 1989.

Penulis berhasil menyelesaikan pendidikan Taman Kanak-kanak di TK Pertiwi, Bandarlampung pada tahun 1994, kemudian dilanjutkan menyelesaikan pendidikan Sekolah Dasar di SDN 02 Teladan, Bandarlampung pada tahun 2000. Sekolah Lanjutan Tingkat Pertama di SLTP Negeri 4 Bandar Lampung pada Tahun 2003 dan Sekolah Menengah Atas di SMA YP UNILA Bandar Lampung pada tahun 2006.

Penulis diterima di Universitas Lampung pada tahun 2006 sebagai mahasiswi jurusan Matematika Program Studi Ilmu Komputer melalui jalur SPMB, dan pada tahun 2009, Penulis melakukan Kerja Praktek di Samudera Indonesia Cabang Panjang.

PERSEMBAHAN

Dengan penuh rasa syukur atas karunia Allah SWT
Ku Persembahkan karya ini untuk:

Bapak & Ibuku Tercinta

Sigit Trenggono, SH & Yenni Rina

Kakak dan Adikku Tersayang

Novita Puspasari

Agustina Tribuana Sari

MOTTO

You can change all things for the better when you change your self for the better

Orang yang paling bijaksana adalah orang yang mengetahui bahwa dirinya tidak tahu
(Socrates)

SANWACANA

Alhamdullillah hirobbil alamin, puji syukur penulis panjatkan kepada Allah SWT, atas berkat limpahan rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi ini.

Skripsi dengan judul " KAMUS BAHASA INGGRIS-INDONESIA DENGAN MENGGUNAKAN JAVA 2 *MICRO EDITION "* adalah salah satu syarat untuk memperoleh gelar Sarjana Komputer di Universitas Lampung.

Dalam kesempatan ini penulis ingin mengucapkan terimakasih kepada :

1. Bapak Akmal Junaidi, M. Sc., selaku pembimbing satu atas kesediaannya memberikan bimbingan, saran, dan kritik sehingga skripsi ini dapat terselesaikan.

2. Ibu Ossy Dwi Endah, M.T., selaku pembimbing dua atas saran yang membangun dalam membimbing penulis.

3. Bapak Dwi Sakethi, M.Kom., selaku pembahas dan penguji skripsi penulis.

4. Bapak Drs. Eri Setiawan., selaku Pembimbing Akademik atas dukungan dan kerjasamanya selama penulis menjadi mahasiswa di Universitas Lampung.
5. Bapak Rangga Firdaus, M.Kom, selaku Ketua Program Studi Ilmu Komputer
6. Bapak Amanto, S.Si.,M.Si., selaku Sekretaris Jurusan Matematika.
7. Bapak Tiryono Rubi, M.Sc.,Ph.D., selaku Ketua Jurusan Matematika.
8. Bapak Dr. Sutyarso, M.Biomed., selaku Dekan FMIPA Universitas Lampung.
9. Bapak dan Ibu Dosen Jurusan Matematika Program Studi Ilmu Komputer.
10. Seluruh Karyawan, Staf, Administrasi Fakultas Matematika dan Ilmu pengetahuan Alam.
11. Seluruh keluargaku tercinta Bapak, Ibu, Mbak Novi, Riri, sepupu-sepupu ku, dan seluruh keluarga besar ku yang tidak dapat disebutkan satu per satu.
12. Oki Putra, terima kasih untuk semua waktu, motivasi, perhatian, nasihat yang membangun dan kebersamaan selama ini.
13. Sahabat sekaligus teman seperjuangan ku Mada, Sani, dan Rere terimakasih banyak ya teman atas semua dukungan semangat dan

bimbingannya selama ini, semoga pertemanan kita akan tetap berlanjut selamanya. Sahabat-sahabat di almamater tercinta Rika, Riska, Joy, Rise, Icha, Restu, Ayu, Meri, Isa, Ajeng, Alif, Lina, Tewe, Valent, Uni, Astri, Rully, Albert, Gogon, Rispab, Tahta, Muis, Bayu, Adit, Satari, Imam, Mayang, Sony, Sueb, Erwin, Dika, Ucup, Ubi, Qiqi, Gani, Iqbal, Jambrong, Boby, Ronal dan semua yang tidak bisa disebutkan satu per satu. yang belum disebut menyusul ya..terima kasih banyak telah menjadi teman yang baik selama ini.

14. Para sahabatku Nina, Winda, Rekha, Bestari, Alin, Iffa, Ayu, Lisha, Debbie, dan Cici. terima kasih telah menjadi sahabat yang tanpa pamrih selama ini, sahabat terbaik untuk ku.

Akhir kata, penulis sadar masih terdapat banyak kekurangan di dalam penulisan skripsi ini, karena terbatasnya kemampuan penulis maupun terbatasnya waktu dan literatur yang mendukung masalah yang dibahas. oleh sebab itu, dengan segala kerendahan hati penulis sangat mengharapkan masukan dan saran yang membangun guna kesempurnaan penulis mendatang.

Bandar Lampung, November 2010
Penulis

Dwi Puspita Sari

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Tuntutan pasar teknologi komunikasi saat ini sangat beragam. Keinginan konsumen yang menggunakan perangkat *mobile* adalah kemudahan dalam

melakukan berbagai aktifitas dengan cepat. Salah satu perangkat *mobile* yang populer adalah telepon selular. Penggunaan telepon selular saat ini sudah menjamur, karena semakin murahnya harga telepon selular. Selain itu semakin banyaknya produsen telepon selular yang mengeluarkan model dan tipe telepon selular terbaru dan dengan fasilitas terbaru yang akhirnya akan berakibat kepada turunnya harga telepon selular keluaran sebelumnya. Saat ini, penggunaan telepon selular tidak lagi terbatas pada alat komunikasi bergerak (*mobile phone*) sebagai pengganti telepon rumah semata, tetapi juga semakin berkembang ke arah penggunaan aplikasi lain seperti memutar video, mendengar musik, ataupun digunakan untuk keperluan khusus seperti kamus.

Teknologi telepon selular sekarang telah menggunakan teknologi Java 2 *Platform Micro Edition* (J2ME) yang memungkinkan pengguna telepon selular menjalankan aplikasi Java. Pengguna dapat membuat program Java yang dapat dijalankan pada telepon selular yang mendukung J2ME *Platform.* Dengan adanya teknologi J2ME, pengguna telepon selular dapat melakukan kreasi sendiri ataupun mengunduh program Java untuk kemudian digunakan pada telepon selularnya. Teknologi J2ME adalah *Platform* yang dikembangkan oleh SUN Microsystem, Inc. yang memungkinkan pengguna telepon selular untuk membuat dan memasang program aplikasi Java pada telepon selularnya.

Dalam penelitian ini, penulis mencoba membuat aplikasi telepon selular berupa aplikasi Java untuk Kamus Bahasa Inggris-Indonesia. Java digunakan sebagai bahasa pengembangan aplikasi ini karena dua alasan. Pertama bahasa Java kompatibel dengan perangkat keras telepon selular. Kedua, fitur bahasa Java yang memiliki kemampuan dalam menciptakan antar muka yang *user friendly.* Dengan kedua alasan ini, Java sangat tepat digunakan untuk menunjang penelitian ini.

1.2 Rumusan Masalah

Masalah yang akan dibahas pada tugas akhir ini adalah bagaimana pengembangan kamus Bahasa Inggris-Indonesia pada telepon selular dengan bahasa Java 2 *Micro Edition.* Selain itu, bagaimana sistem kamus dirancang agar menghasilkan aplikasi yang mudah di*install* dan dipakai serta berpeluang untuk dikembangkan pada masa yang akan datang.

1.3 Tujuan Tugas Akhir

Tujuan dari pembuatan tugas akhir ini adalah :

1. Untuk membuat kamus (*dictionary*) yang memungkinkan pengguna telepon selular mencari kata dalam Bahasa Inggris maupun Bahasa Indonesia. Kamus dapat digunakan dengan memanfaatkan telepon selular yang berfungsi sebagai *mobile dictionary.*

2. Merintis penggunaan J2ME sebagai aplikasi untuk perangkat *mobile* seperti telepon selular, PDA, dan sebagainya.

1.4 Batasan Masalah

Batasan masalah yang digunakan dalam penelitian tugas akhir ini adalah sebagai berikut :

a. Penulis hanya membahas tentang aplikasi Kamus Bahasa Inggris-Indonesia dengan menggunakan teknologi Java 2 *Micro Edition* (J2ME).
b. *Database* yang akan digunakan untuk mendukung aplikasi ini adalah *Record Management System* (RMS).

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari aplikasi yang dihasilkan dalam penelitian ini adalah :

a. Memfasilitasi *user* yang ingin mencari kata-kata Indonesia dan /atau Inggris melalui *mobile dictionary* pada telepon selular.
b. Dapat dijadikan referensi bagi yang ingin mendalami pemrograman J2ME (Java 2 *Micro Edition*).

BAB II

LANDASAN TEORI

2.1 Perangkat *Mobile*

Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, desain dan *layout*, tetapi mereka memiliki kesamaan karakteristik yang sangat berbeda dari sistem *desktop*. (JENI, 2010). Perbedaannya antara lain adalah :

a. Ukuran yang kecil

Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

b. Memori yang terbatas

   Perangkat *mobile* juga memiliki memori yang kecil, yaitu *Primary* (RAM) dan *secondary* (*disk*). Pembatasan jumlah memori ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis perangkat *mobile*.

c. Daya proses yang terbatas

   Sistem *mobile* tidaklah *desktop*. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi kinerja dari sumber daya ini.

d. Mengkonsumsi daya yang rendah

   Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan *desktop*. Perangkat ini harus menghemat daya karena mereka berjalan pada keadaan dimana daya yang disediakan dibatasi oleh baterai.

e. Kuat dan dapat diandalkan

   Karena perangkat *mobile* selalu dibawa kemana saja, perangkat *mobile* harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sedikit tetesan-tetesan air.

f. Konektivitas yang terbatas

Perangkat *mobile* memiliki *bandwith* rendah bahkan beberapa dari perangkat ini bahkan tidak memiliki atau menggunakan *bandwidth.* Konektifitas perangkat seperti ini umumnya bergantung pada media *wireless*.

g. Masa *booting* yang singkat

Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala. Contoh beberapa kasus adalah sebuah telepon selular. Telepon selular dapat melakukan *booting* dalam hitungan detik.

2.2 Sejarah Java

Java bermula dari proyek penelitian perusahaan Sun Microsystems dengan nama sandi Green pada tahun 1991. Terdapat prediksi bahwa miroprosesor akan digunakan luas pada peralatan-peralatan elektronik (Hakim et al, 2009). Oleh karena adanya bermacam tipe mikroprosesor, maka dibutuhkan sebuah bahasa pemrograman yang dapat berjalan di semua mikroprosesor. Pada tahun 1995, Sun Microsystems mengumumkan kehadiran bahasa Java secara formal kepada masyarakat luas seiring dengan meledaknya era internet. Bahasa pemrograman Java pertama kali diperkenalkan oleh James Gosling, yaitu salah seorang yang berperan besar dalam proyek bahasa pemrograman ini.

2.2.1 Perkembangan Teknologi Java

Saat ini Sun Microsystems membagi Java menjadi empat jenis edisi : (Raharjo et al, 2010).

a. Java Card : Teknologi Java yang digunakan pada peralatan elektronik yang memiliki memori yang sangat terbatas, misalnya *smart card*.
b. J2ME : Java 2 *Platform, Micro Edition,* merupakan teknologi Java edisi mikro, digunakan untuk penerapan teknologi Java pada peralatan elektronik kecil seperti telepon selular dan PDA.
c. J2SE : Java 2 *Platform, Standard Edition,* merupakan teknologi Java edisi standar, digunakan untuk penerapan teknologi Java pada komputer *desktop.*
d. J2EE : Java 2 *Platform, Enterprise Edition,* merupakan teknologi Java edisi enterprise, digunakan untuk penerapan teknologi Java pada peralatan komputer *desktop* ataupun *server*.

Setiap edisi Java terdiri dari dua komponen utama berikut :

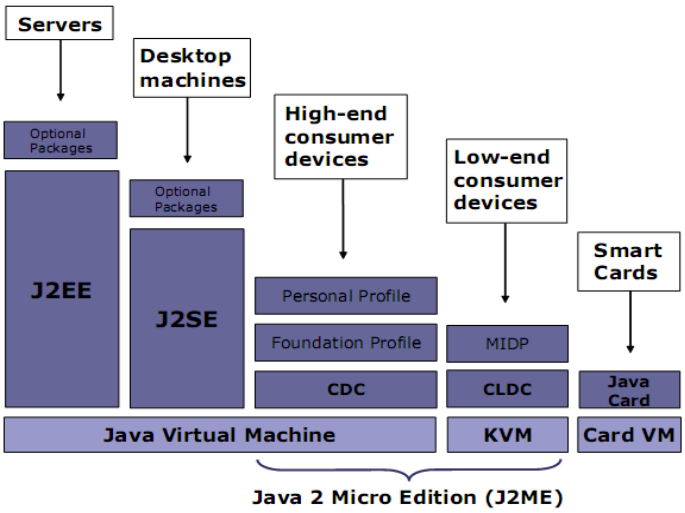
a. Java *Application Programming Interface* (Java API).

   Java API terdiri atas kumpulan *library* yang digunakan untuk keperluan pemrograman. Dengan adanya API, seorang *programmer* tidak harus membuat program dari awal. Misalnya pada J2SE, librari-nya telah dilengkapi API

untuk pengolahan *window* dengan Swing dan AWT (*Abstract Window Toolkit*).

b. Java *Run Time Environment* (JRE).

JRE merupakan lingkungan yang membuat aplikasi Java dapat dijalankan. Salah satu komponen penting JRE adalah Java *Virtual Machine* (JVM).



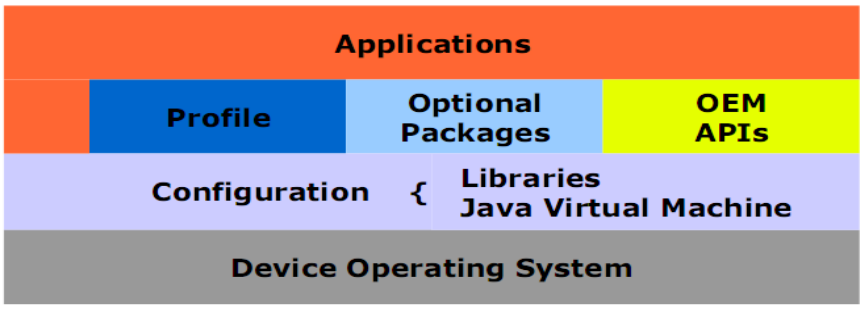
Gambar 2.1 *Platform Java*

Seperti pada gambar 2.1 di atas, J2ME pada dasarnya terdiri dari tiga buah bagian, yaitu konfigurasi, profil, dan paket-paket opsional. Dua buah konfigurasi yang disediakan oleh Sun Microsystems, yaitu CLDC (*Connected Limited Device Conviguration*) dan CDC *(Connected Device Configuration)*. Sedangkan dalam profil telah disediakan pula profil yang diberi nama dengan MIDP (*Mobile Information Device Profile).*

2.3 Java 2 *Micro Edition* (J2ME)

Java 2 *Micro Edition* (J2ME) adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen. Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya baterai, layar yang kecil dan *bandwidth* jaringan yang rendah. (JENI, 2010). Dengan perkembangan perangkat *mobile* konsumer dari telepon, PDA, kotak permainan ke peralatan-peralatan rumah, Java menyediakan suatu lingkungan yang *portable* untuk mengembangkan dan menjalankan aplikasi pada perangkat ini.

Program J2ME, seperti semua program Java adalah diterjemahkan oleh VM (*Virtual Machine)*. Program-program tersebut di *compile* ke dalam *bytecode* dan diterjemahkan dengan Java *Virtual Machine* (JVM). Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat keras. J2ME menyediakan suatu *interface* yang sesuai dengan perangkat. Aplikasi-aplikasi tersebut tidak harus di *compile* ulang supaya mampu dijalankan pada mesin yang berbeda. Inti penggunaan dari J2ME terletak pada *configuration* dan profil-profil. Suatu *configuration* menggambarkan lingkungan *runtime* dasar dari suatu sistem J2ME.



Gambar 2.2 Arsitektur J2ME

Sebuah profil memberikan *library* tambahan untuk suatu kelas tertentu pada sebuah perangkat. *Profile-profile* menyediakan *user interface* (UI) API, *persistence*, *messaging library*, dan sebagainya.

Satu set *library* tambahan atau *package* tambahan menyediakan kemampuan program tambahan. Pemasukan *package* ini ke dalam perangkat J2ME dapat berubah-ubah tergantung pada kemampuan perangkat yang digunakan. Sebagai contoh, beberapa perangkat MIDP tidak memiliki *Bluetooth built-in*, sehingga *Bluetooth* API tidak disediakan dalam perangkat ini.

2.3.1 *Configuration*

Suatu *configuration* menggambarkan fitur minimal dari lingkungan lengkap Java Runtime namun bukan menggambarkan fitur tambahan (Raharjo et al, 2010). C*onfiguration* ini digunakan untuk menjamin kemampuan portabilitas dan interoperabilitas optimal di antara berbagai macam perangkat yang dibatasi sumber dayanya (*memory*, prosesor, koneksi yang dibatasi). Suatu *configuration* J2ME

menggambarkan suatu komplemen yang minimum dari teknologi Java. *configuration* menggambarkan:

a) Subset bahasa pemrograman JAVA

b) Kemampuan Java *Virtual Machine*(JVM)

c) *Core platform libraries*

d) Fitur sekuriti dan jaringan

2.3.2 *Profile*

Suatu *profile* menggambarkan set-set tambahan dari API dan fitur untuk pasar tertentu, kategori perangkat atau industri. *Profile-profile* menggambarkan *library* yang penting untuk membuat aplikasi-aplikasi menjadi lebih efektif. *Library* ini memasukkan *user interface*, jaringan dan penyimpanan API.

2.4 CLDC

The Connected Limited Device Configuration (CLDC) menggambarkan dan menunjuk pada area berikut ini:

a. Fitur Bahasa Java dan *Virtual Machine* (VM)
b. *Library* dasar (java.lang.*,java.util.*)
c. *Input/Output* (java.io.*)
d. Keamanan
e. Jaringan
f. *Internationalization*

2.4.1 Fitur yang Hilang

Fitur tertentu dari J2SE yang dipindahkan dari CLDC adalah :

a. Finalization of class instances

b. Asynchronous exceptions

c. Beberapa *error classes*

d. User-defined class loaders

e. Reflection

f. Java *Native Interface* (JNI)

g. *Thread groups* dan *daemon threads*

Reflection, Java *Native Interface* (JNI) dan *user-defined class loaders* potensial menjadi lubang keamanan. JNI juga membutuhkan memori yang intensif sehingga dimungkinkan untuk tidak mendapat dukungan dari memori rendah sebuah perangkat *mobile*.

2.4.2 Karakteristik Perangkat CLDC

Perangkat yang dibutuhkan oleh CLDC mempunyai karakteristik sebagai berikut:

a) Memori minimal 192 kb untuk *platform* Java.

b) Prosesor dengan 16 atau 32 bit.

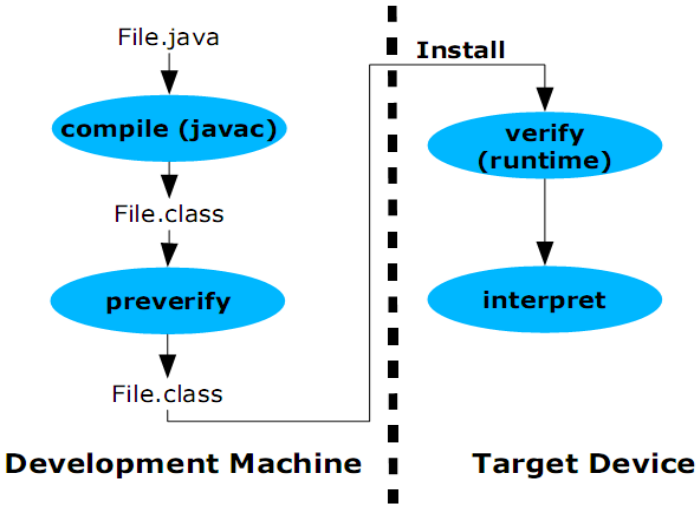
c) Mengkonsumsi daya yang kecil.

d) Koneksi jaringan yang sementara dengan pembatasan *bandwidth* (biasanya *wireless*).

Profile yang berada di bawah CLDC menggambarkan instalasi dan daur hidup sebuah aplikasi, antarmuka (UI) dan penanganan peristiwa (*event handling*).

2.4.3 Verifikasi Class

Spesifikasi CLDC memerlukan semua *class* untuk melewati proses verifikasi dua tingkat. Verifikasi pertama dilaksanakan diluar perangkat sebelum instalasi pada perangkat. Verifikasi kedua terjadi pada perangkat selama *runtime* dan dilaksanakan oleh KVM.



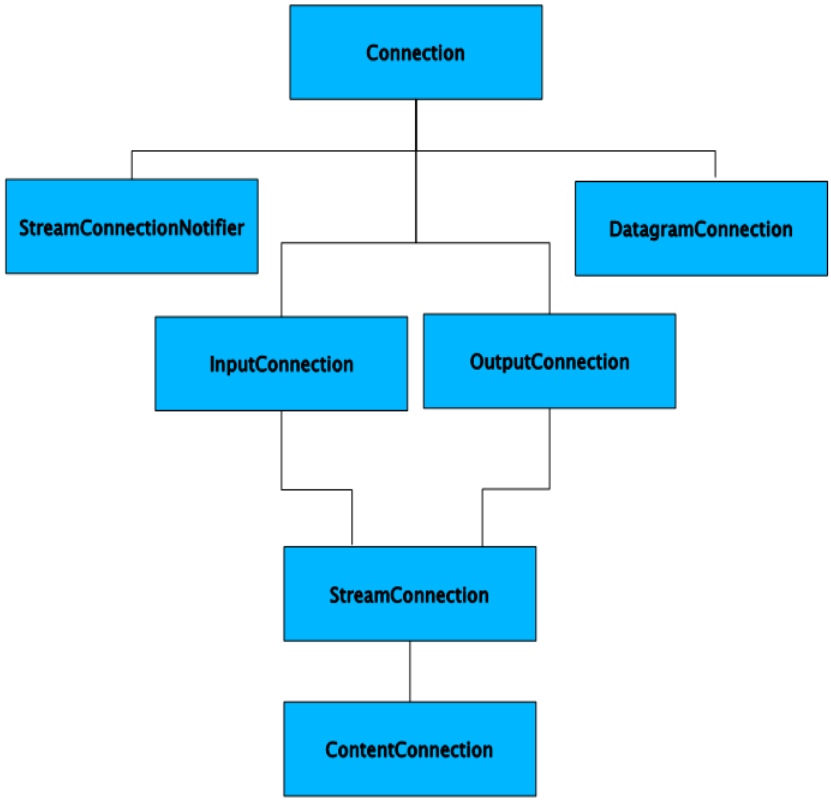
Gambar 2.3 Proses verifikasi dua tingkat

2.4.4 *Generic Connection Frameworks*

The Generic Connection Framework menyediakan API dasar untuk koneksi dalam CLDC. Framework ini menyediakan suatu pondasi

umum untuk koneksi seperti *HTTP*, *Socket*, dan *Datagrams* . (JENI, 2010). GCF menyediakan suatu set API yang umum dan biasa yang memisahkan semua jenis koneksi. Tidak semua jenis koneksi dibutuhkan untuk diterapkan oleh perangkat MIDP.

Hirarki *interface* yang dapat diperluas dari GFC membuat proses penyamarataan menjadi mungkin. Jenis koneksi baru mungkin bisa ditambahkan ke dalam *framework* ini dengan memperluas hirarki *interface* ini.



Gambar 2.4 Hirearki koneksi GCF

2.5 CDC

Connected Device Configuration (CDC) adalah super set dari CLDC. CDC menyediakan lingkungan Java *runtime* yang lebih luas dibandingkan CLDC dan lebih dekat kepada lingkungan J2SE.

CDC Java *Virtual Machine* (CVM) mendukung penuh Java *Virtual Machine* (JVM). CDC berisi semua API dari CLDC. CDC menyediakan suatu subset yang lebih besar dari semua *class* J2SE. Seperti CLDC, CDC tidak menggambarkan setiap *class* UI. *Library* UI digambarkan oleh *profile- profile* di bawah *configuration* ini.

Semua *class* yang terdapat dalam CDC berasal dari *package* ini:

a. java.io
b. java.lang
c. java.lang.ref
d. java.lang.math
e. java.net
f. java.security
g. java.security.cert
h. java.text
i. java.util
j. java.util.jar
k. java.util.zip

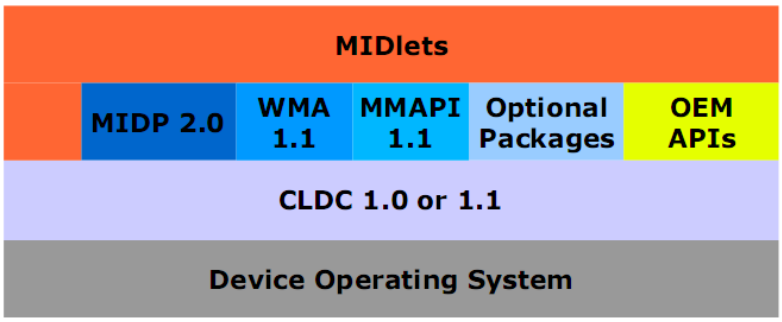
CDC juga memasukkan GCF (*Generic Connection Framework*) di

dalamnya untuk menyediakan pondasi umum dari berbagai koneksi, seperti *HTTP, socket,* dan *datagram*.

2.6 JTWI

The Java Technology for the Wireless Industry (JTWI) menetapkan satu set jasa dan spesifikasi standar industri *wireless.* (Hakim et al, 2010). Berdasarkan spesifikasi, fasilitas yang disediakan oleh JTWI adalah "untuk memperkecil fragmentasi API di dalam pasar telepon selular, dan untuk mengirim spesifikasi yang dapat diprediksi, spesifikasi yang jelas untuk perangkat pabrik, operator, dan pengembang aplikasi".

Dengan penyesuaian pada JTWI, banyak aplikasi akan berjalan di perangkat yang lebih luas dan beragam. Perangkat pabrik juga akan beruntung karena dengan JTWI spesifikasi standar akan tersedia untuk perangkat tersebut.



Gambar 2.5 Komponen JTWI

2.7 MIDP

The Mobile Information Device Profile (MIDP) berada di atas dari CLDC. Seorang *programmer* tidak dapat menulis aplikasi *mobile* hanya dengan menggunakan CLDC API. Program yang dikembangkan tetap memanfaatkan MIDP untuk mendefinisikan UI. (Johannes, 2010). Spesifikasi MIDP, kebanyakan seperti CLDC dan API lainnya sudah digambarkan melalui Java *Community Process* (JCP). JCP melibatkan sebuah kelompok ahli berasal dari lebih dari 50 perusahaan, yang terdiri atas pabrik perangkat *mobile*, pengembang *software.* MIDP terus berkembang, dengan versi-versi masa depan yang telah lulus dari proses ketat JCP.

Spesifikasi MIDP menggambarkan suatu perangkat MID (*Mobile Information Device)* yang memiliki karakteristik-karateristik ini sebagai batas minimum:

a. Tampilan:
   i) Ukuran Layar: 96x54 piksel
   ii) Kedalaman tampilan: 1-bit
   iii) Ketajaman pixel: sekitar 1:1
b. Masukan: Satu atau lebih mekanisme *user-input*, satu *keyboard*, dua *keyboard*, atau *touchscreen.*
c. Memori: 256 *kilobytes of non-volatile* memori untuk implementasi MIDP.
d. 8 *kilobytes of non-volatile* memori for *application-created*

persistent data

e. 128 kilobytes *of volatile memory for the* Java runtime (e.g., *the Java heap*)
f. Jaringan: Dua jalur, *wireless, bandwidth* terbatas
g. *Sound*: Kemampuan untuk memainkan nada-nada

MIDP menggambarkan model aplikasi, *User Interface* API, penyimpanan dan jaringan yang kuat, permainan dan media API, kebijakan keamanan, penyebaran aplikasi dan ketetapan *over-the-air*.

2.8 MIDlet

Suatu aplikasi MIDP disebut MIDlet. Perangkat *Application Management Software* (AMS) berinteraksi langsung dengan MIDlet dengan method MIDlet *create, start, pause,* dan *destroy.*

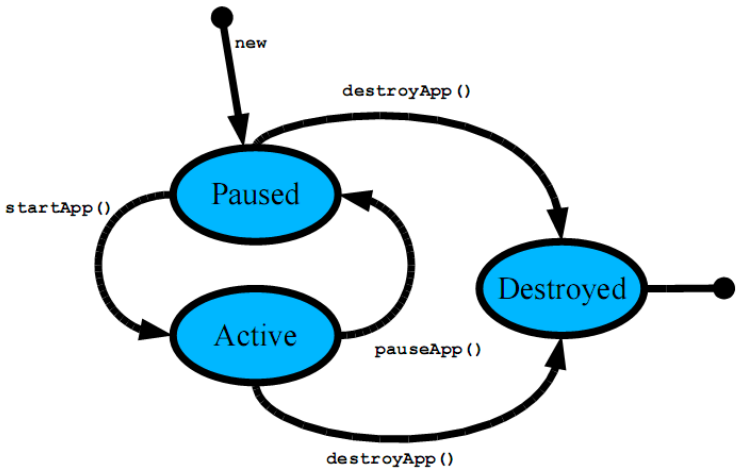
MIDlet adalah bagian dari package javax.microedition.midlet. Sebuah MIDlet harus di*extend* dengan *class* MIDlet dan dapat meminta parameter dari AMS seperti dirumuskan dalam *Application Descriptor* (JAD). Suatu MIDlet tidak harus memiliki sebuah *method* public static void main (String [] argv), *method* tersebut tidak akan dikenal lagi oleh AMS sebagai titik awal suatu program.

2.8.1 Siklus MIDlet

Siklus hidup MIDlet dimulai ketika di*instantiate* oleh AMS. Setelah

perintah inisiasi dieksekusi MIDlet pada awalnya masuk status "*Pause*" setelah perintah baru dibuat. AMS memanggil *constructor public* tanpa argumen dari MIDlet. Jika sebuah *exception* terjadi dalam *constructor*, MIDlet memasuki status "*Destroyed*" dan membuangnya segera. MIDlet masuk ke dalam status "*Active*" atas pemanggilan *method* startUp() oleh AMS.

MIDlet masuk ke dalam status "*Destroyed*" ketika AMS memanggil *method* destroyApp(). Status ini juga kembali diakses ketika *method* notifyDestroyed() kembali dengan sukses kepada aplikasi. Dengan catatan bahwa MIDlet hanya bisa memasuki status " *Destroyed*" sekali dalam siklus masa hidupnya.



Gambar 2.6 Daur Hidup MIDlet

2.8.2 MIDlet *Suites*

Aplikasi-aplikasi MIDlet dibungkus dan dikirim ke dalam perangkat sebagai MIDlet *suites*. Sebuah MIDlet *suite* terdiri dari Java Archive (JAR) dan sebuah tambahan Java *Application Descriptor* (JAD). File JAD adalah suatu file teks yang berisi satu set atribut-atribut.

2.9 Database (*Record Management System*)

Salah satu fitur utama dari MIDP adalah RMS (*Record Management System*), sebuah API yang memberikan MIDP *application* kemampuan untuk menyimpan data di dalam perangkat selular. (Johannes, 2010). Hampir semua perangkat MIDP saat ini hanya mendukung RMS sebagai satu-satunya media penyimpanan data. Oleh karena itu RMS sangat penting dipahami untuk mengembangkan aplikasi di perangkat selular.

Dengan menggunakan RMS, data-data seperti data pesanan, data pelanggan dan sebagainya dapat disimpan di perangkat selular seperti layaknya penyimpanan data menggunakan basis data konvensional. Ada dua konsep penting dari RMS, yaitu :

a. Record Store

Record store merupakan kumpulan dari *record-record*. Setiap *record* harus berada pada satu *record store* dan semua akses ke *record* harus melalui *record store*. Ketika sebuah *record* dibuat, *record store* akan memberikan *record* tersebut sebuah nomor identifikasi yang unik (*id record*). Nomor identifikasi tersebut

dimulai dari 1 dan maksimal 2.187.483.647. Dengan menganalogikan basis data konvensional, *record store* merupakan tabel dan *record* merupakan *row* yang ada di tabel tersebut.

b. Record

Record adalah sebuah data tunggal. *Record* dapat berupa *numeric, string, array* ataupun gambar selama data itu dapat diwakilkan dalam bentuk rangkaian *bytes*. Berbeda dengan basis data konvensional dimana dalam satu *record* terdiri dari beberapa *field*, *record* di RMS tidak mempunyai banyak *field* melainkan hanya 1 buah *field* yang berisi *array of byte*. RMS hanya menyediakan media penyimpanan dan nomor identifikasi yang unik untuk setiap *record* agar aplikasi dapat mengidentifikasikan *field-field* di dalam *array of byte* tersebut itu semua tergantung dari implementasi masing-masing aplikasi.

2.9.1 Mengenal Data Presisten

Dalam pemrograman MIDP, terdapat tiga buah ruang penyimpanan data yang dapat digunakan, yaitu : *volatile* RAM, ruang presisten, dan penyimpanan secara *remote*. (Raharjo et al, 2010).

2.9.1.1 *Volatile* RAM

RAM (*Random Access Memory*) menyediakan akses yang sangat cepat terhadap data-data yang digunakan oleh aplikasi, sama seperti pada saat sebuah objek dibuat dalam aplikasi. Meskipun demikian,

RAM bersifat volatil atau penyimpanannya bersifat sementara. Data – data yang disimpan di dalam RAM akan hilang ketika *device* dimatikan atau aplikasi MIDlet yang kita buat ditutup atau dihentikan.

Dalam terminologi Java, pengaturan RAM sudah diatur oleh JVM (Java *Virtual Machine*) dengan menghadirkan *garbage collector*. Konsep ini merupakan suatu mekanisme mengenai objek-objek yang sudah tidak diacu atau tidak diperlukan lagi akan dibuang dari memori secara otomatis, tanpa adanya intervensi atau campur tangan dari *programmer.*

2.9.1.2 Ruang Presisten

Ruang presisten adalah ruang memori lokal yang terdapat di dalam *device,* sama halnya seperti *flash memory* atau *hard drive.* Tidak seperti RAM, ruang persisten dapat menyimpan data-data dalam aplikasi secara permanen. Ini artinya, meskipun *device* dimatikan atau aplikasi ditutup, data masih tersimpan di memori, dan dapat diambil atau dibuka lagi pada saat *device* dinyalakan atau aplikasi dibuka kembali. Ruang presisten harus diatur sendiri secara efisien oleh *programmer*, bukan diatur langsung oleh JVM.

Untuk dapat melakukan penyimpanan data lokal secara *default,* kita dapat menempatkan data tersebut sebagai atribut dari *file* JAD atau

JAR yang telah dibuat. Data yang tersimpan dengan cara ini nilainya bersifat *read-only* atau hanya dapat dibaca sehingga tidak dapat mengganti nilai tersebut pada saat aplikasi dijalankan. Pembacaan nilai dari atribut yang dimasukkan ke dalam *file* JAD atau JAR, dapat memanfaatkan *method* getAppProperty (), yang memiliki bentuk deklarasi sebagai berikut :

Public String getAppProperty (string key)

Dalam deklarasi ini parameter *key* adalah nama atribut yang akan diambil nilainya.

2.9.1.3 Ruang Penyimpanan *Remote*

MIDlet yang terhubung dengan suatu jaringan internet dapat mengakses atau menyimpan data pada ruang penyimpanan *remote* (misalnya :sebuah server *database*) melalui jaringan tersebut. Dalam aplikasi yang kompleks, terkadang perlu dilakukan kombinasi antara ruang penyimpanan lokal (di dalam *device*) dan ruang penyimpanan *remote* (biasanya terdapat pada sebuah *server web*).

2.9.2 Serialisasi Objek Java

Pada saat data disimpan di dalam sebuah *record store*, data tersebut

berada dalam bentuk sekumpulan larik dari tipe *byte (array byte).* Namun sebaliknya, di dalam pemrograman Java, di sisi aplikasi, data harus selalu dalam bentuk objek. Karena alasan itulah, untuk melakukan penyimpanan data dari aplikasi ke dalam *record store*, terlebih dahulu harus dilakukan serialisasi dari bentuk objek ke bentuk *array byte.* Untuk mengembalikan data dari *record store* ke aplikasi, perlu dilakukan juga deserialisasi dari bentuk *array byte* ke bentuk objek. Dalam pemrograman MIDP, tidak didukung adanya fasilitas untuk serialisasi atau deserialisasi untuk objek-objek Java yang terdapat di dalam J2SE. Untuk itu, maka perlu dilakukan proses tersebut secara manual melalui kode-kode program yang ditulis, melalui kelas-kelas yang tersimpan pada paket java.io. Sebagian besar kelas yang terdapat dalam paket tersebut digunakan untuk merepresentasikan data dalam bentuk *stream.*

2.9.3 Kelas RecordStore

Kelas RecordStore adalah satu-satunya kelas konkrit yang terdapat dalam paket javax.microedition.rms. Kelas tersebut digunakan untuk membuat *record store* di dalam aplikasi. Sebuah *record store* akan berisi sekumpulan *record* yang diindentifikasi dengan suatu ID yang bersifat unik. ID tersebut akan berperan sebagai *primary key* dari *record-record* yang dimasukkan sehingga dalam aplikasi dapat memiliki dua atau lebih *record* dengan ID yang sama.

2.9.3.1 Memanipulasi *Record Store*

Untuk dapat bekerja dengan *record store,* hal yang pertama kali harus dilakukan adalah membuat objek dari kelas RecordStore. Objek tersebut dapat dibuat dengan menggunakan *method* openRecordStore() yang didefinisikan dalam kelas RecordStore yang merupakan *method* statik. Berikut ini bentuk umum deklarasi *method* tersebut:

Static RecordStore openRecordStore (String name, booleancreate)

2.9.3.1.1 Menambah *Record* Baru ke dalam *Record Store*

Untuk menambah *record* ke *record store* digunakan perintah addRecord(). (Johannes, 2010). Perintah addRecord() mempunyai tiga parameter, yaitu :

a. Data (:byte[]) : byte *array* dari objek yang ingin disimpan
b. Offset (:int) : Offset dari data
c. numBytes (:int) : Jumlah byte dari data atau besar dari byte *array* yang disimpan.

Perintah addRecord() mempunyai nilai kembali, berupa :

a. RecordId (:Int) : id *record* yang diberikan untuk *record* tersebut.

2.9.3.1.2 Menghapus *Record* dari *Record Store*

Untuk menghapus *record* dari *record store* digunakan perintah deleteRecord(). Perintah deleteRecord() mempunyai 1 parameter, yaitu : recordId (:int) : id dari *record* yang ingin dihapus. Contoh penggunaan : rs.deleteRecord(rid);

2.9.3.1.3 Mengakses dan Membaca *Record-record* dari *Record Store*

Untuk mengakses dan membaca *record-record* dari *record store* digunakan perintah enumerateRecord() dari *interface* RecordEnumeration. Perintah enumerateRecord() mempunyai tiga parameter,yaitu :

a. Filter (:RecordFilter) : kriteria untuk pemilihan *record*
b. Comparator (:RecordComparator) : kriteria untuk pengurutan *record*
c. keepUpdate (:boolean)

Record store juga menyediakan fasilitas bagi *programmer* untuk menggunakan kriteria pemilihan dan kriteria pengurutan *record-record.* Hal ini dilakukan dengan mengimplementasikan *interface* RecordFilter dan RecordComparator.

2.9.3.2 RecordFilter

RecordFilter merupakan *method* yang memungkinkan *programmer* untuk memasukkan kriteria pemilihan *record.* Untuk menggunakan

RecordFilter, suatu program harus mengimplementasikan perintah matches(). *Method* matches() mempunyai satu parameter, yaitu :

Rec (:byte[]) : byte *array* dari objek yang ingin dibandingkan dengan kriteria yang diinginkan.

Method matches() mempunyai nilai kembali, berupa :

(:boolean) : apakah *record* yang dibandingkan memenuhi kriteria atau tidak.

2.9.3.3 RecordComparator

RecordComparator memungkinkan *programmer* untuk memasukkan kriteria pengurutan *record.* Untuk menggunakan RecordComparator, suatu program harus mengimplementasikan perintah compare(). Perintah Compare() mempunyai dua parameter, yaitu :

a. Rec1 (:byte[]) : byte *array* dari objek pertama
b. Rec2 (:byte[]) : byte *array* dari objek selanjutnya

Perintah compare() mempunyai nilai kembali, berupa :

(:int) : perbandingan antara rec1 dengan rec2.

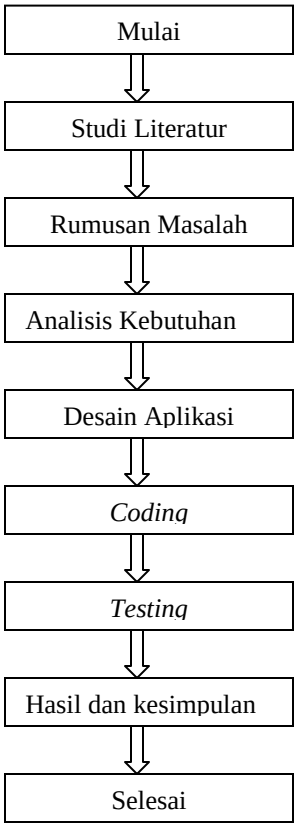
- RecordComparator.FOLLOWS : apabila rec1>rec2
- RecordComparator.PRECEDES: apabila rec1<rec2
- RecordComparator.EQUIVALENT : apabila rec1=rec2

BAB III

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Dalam penulisan tugas akhir ini terdapat alur metodologi penelitian seperti yang terdapat pada gambar di bawah ini :



Gambar 3.1 Alur Metodologi Penelitian

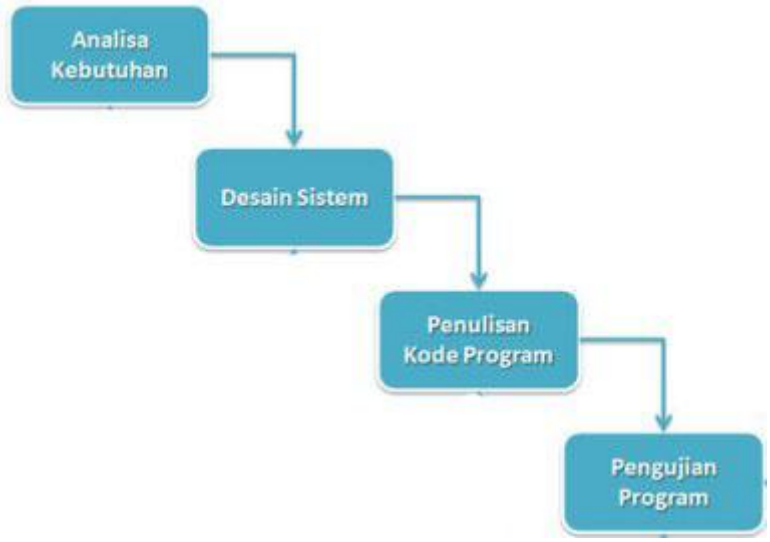
3.1.1 Metode Pengumpulan Data

Metode pengumpulan data yang akan dilakukan adalah metode penelitian kepustakaan. Secara teknis metode ini dilaksanakan dengan cara mengumpulkan data dan informasi yang diperoleh dari buku dan analisis perancangan sistem, buku pemrograman, artikel dari internet, maupun sumber informasi lain yang relevan dengan pembahasan pada penelitian ini.

3.1.2 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan pada perancangan sistem aplikasi kamus Bahasa Inggris – Indonesia ini adalah *Linear Sequential Model* atau sering disebut Metode Waterfall.

Metode ini mengusulkan sebuah pendekatan kepada perkembangan perangkat lunak yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem pada seluruh tahapan analisis, perancangan, kode, pengujian, dan pemeliharaan. Metode ini memiliki empat tahapan, yaitu tahap analisis, perancangan, kode, dan pengujian. Berikut skema dari metode Waterfall.



Gambar 3.2 *Linear Sequential Model* (Pressman, 2001:29)

1. Analisa Kebutuhan

Tahap analisis adalah proses untuk mengumpulkan kebutuhan yang diperlukan oleh aplikasi.

2. Desain Sistem

   Tahap desain sistem adalah proses dimana kebutuhan yang telah didapat pada tahap analisis ditransformasikan menjadi model sistem sehingga mudah dibuat menjadi aplikasi yang sesuai dengan kebutuhan *user*. *Tool* yang digunakan untuk merancang desain sistem adalah dengan menggunaka fasilitas *insert*, *picture*, *autoshapes* yang terdapat dalam *microsoft word.*

3. Penulisan Kode Program

   Tahap ini merupakan tahap di mana perancangan yang telah dilakukan ditransformasikan melalui proses *coding* dengan bahasa pemrograman Java sehingga terbentuk aplikasi kamus Bahasa Inggris-Indonesia.

4. Pengujian Program

   Pengujian akan dilakukan dengan menggunakan metode *blackbox*, yaitu pengujian dilakukan tanpa melihat kode program.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Analisa Kebutuhan

Berdasarkan hasil analisis secara umum, ada beberapa hal yang dibutuhkan dalam pemrosesan kerja sistem, yaitu :

a. Kebutuhan Sistem

Analisis kebutuhan sistem meliputi 2 (dua) hal, yaitu : *Hardware* (Perangkat keras) dan *Software* (perangkat lunak).

1. Perangkat Keras (*Hardware*)

Spesifikasi *hardware* yang digunakan untuk mengembangkan sistem ini yaitu :

i. *Processor* Intel Celeron atau lebih

ii. Kapasitas *Random Access Memory* (RAM) 512 MB

iii. *Harddisk* 120 GB

iv. Telepon selular yang telah mendukung Java MIDP, dalam kasus ini penulis menggunakan produk Nokia 6303, Blackberry 8900, dan Samsung B3310.

2. Perangkat Lunak (*Software*)

Spesifikasi *Software* yang digunakan untuk mengmbangkan sistem ini yaitu :

i. Sistem Operasi *Microsoft Windows* XP

ii. Program Aplikasi *Netbeans* IDE 6.

b. Kebutuhan Pengguna

Berdasarkan dari kegiatan yang telah dilakukan melalui studi literatur dan membaca referensi yang diperoleh dalam menganalisis kebutuhan pengguna untuk menerapkan fungsi-fungsi dari aplikasi yang akan dibuat, terdiri dari 2 (dua) menu utama, antara lain :

1. Menu awal

a. Menu tambah kata

i. Masukkan kata

ii. Arti

b. Menu Daftar Kata

c. Menu Pencarian Kata

2. Menu Tombol

a. Pilih

b. Simpan

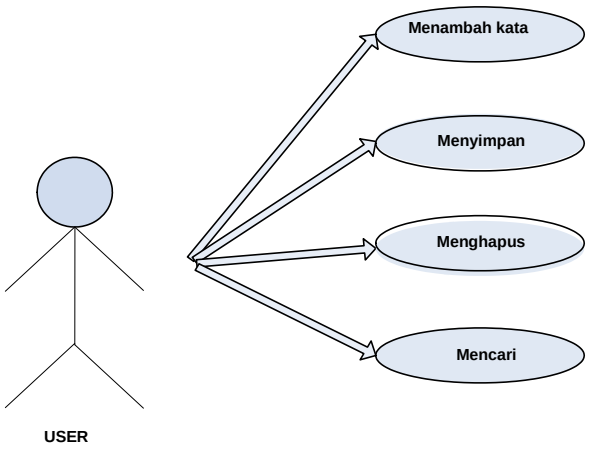
c. Kembali

d. Hapus

e. Pencarian

f. Keluar

c. *Use Case* Diagram



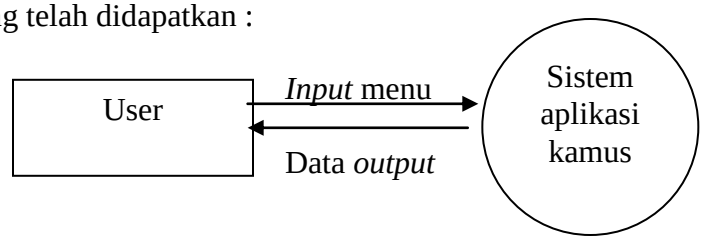
Gambar 4.1 *Use Case*

Seperti yang terlihat pada *use case* di atas, dalam aplikasi ini, *user* dapat melakukan beberapa hal, seperti menambah kata, menyimpan kata,

menghapus kata, serta mencari kata.

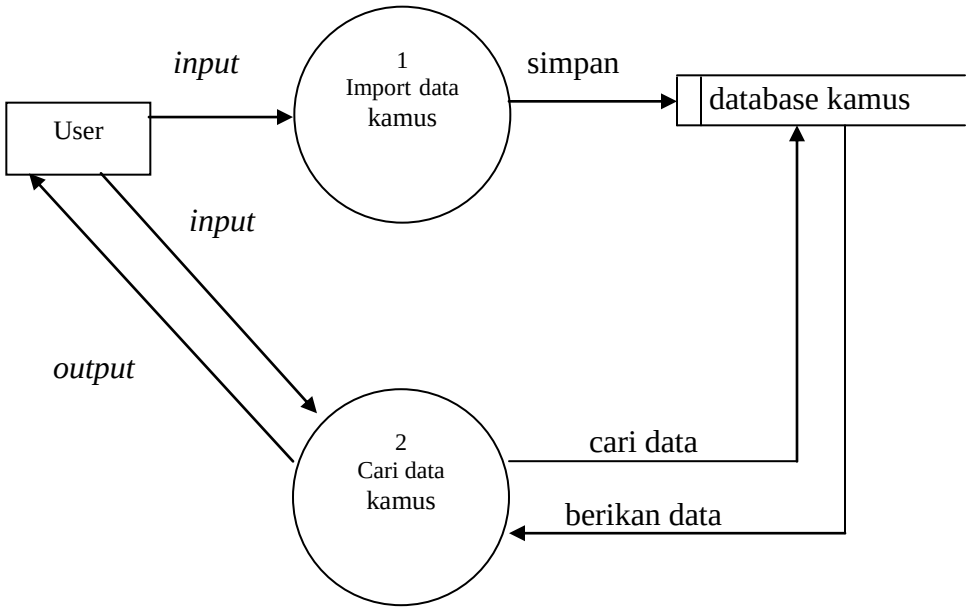
4.2 Desain Sistem

Desain sistem yang akan ditampilkan dalam penelitian ini berupa *form* sederhana yang nantinya akan dibuat menggunakan bahasa pemrograman Java GUI sebagai *user interface*. Berikut adalah diagram konteks dari data yang telah didapatkan :



Gambar 4.2 Diagram Konteks Kamus Bahasa Inggris-Indonesia

Diagram konteks di atas menggambarkan proses secara umum dari aplikasi yang dibuat. Penjelasannya adalah, *user* melakukan *input* pada aplikasi, kemudian aplikasi memproses *input*an tersebut sehingga menghasilkan data *output* untuk diperlihatkan pada *user*.

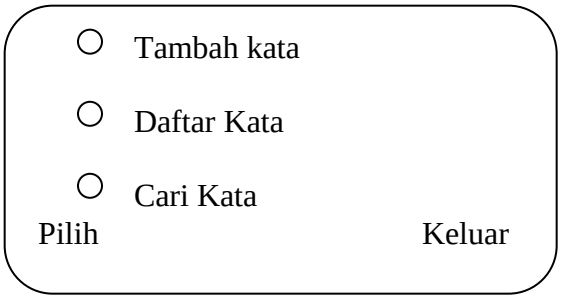


Gambar 4.3 Diagram level 0 Kamus Bahasa Inggris-Indonesia

Diagram level 0 di atas menggambarkan *user* melakukan *input* pada aplikasi, selanjutnya aplikasi akan menyimpan ke dalam *database* kamus. Selain itu, *user* juga dapat mencari data dalam kamus. *User* meng*input*kan data, maka proses pencarian akan dilakukan dalam *database* kamus. Setelah ditemukan, data menghasilakan *output* dan diperlihatkan kepada *user* .

Berikut ini adalah rancangan desain *form* menu utama, menambah kata, mendaftarkan kata, dan menu pencarian kata :

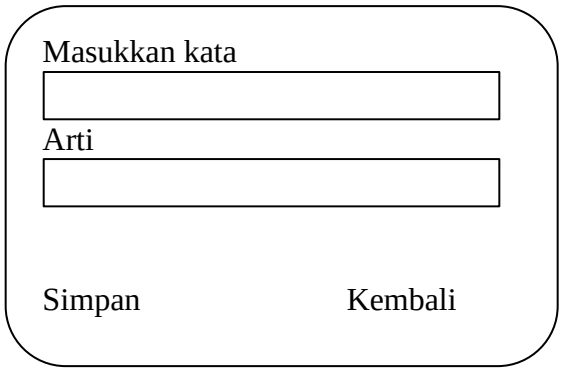
a. Menu Utama



Gambar 4.4 Desain menu utama

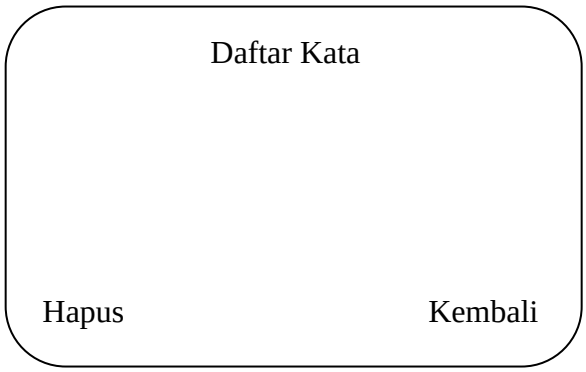
Di dalam menu utama, menu yang pertama kali akan ditampilkan adalah menu tambah kata, daftar kata, dan cari kata. Tombol pilih digunakan untuk memilih menu yang tersedia, sedangkan tombol keluar digunakan untuk keluar dari aplikasi kamus.

b. Menu Tambah Kata



Jika memilih menu tambah kata, maka tampilan yang akan dihasilkan adalah masukkan kata dan arti, selanjutnya tombol simpan yang terdapat pada sebelah kiri digunakan untuk menyimpan kata yang telah diketikkan ke dalam *database*. Tombol kembali pada sebelah kanan bawah digunakan untuk kembali ke menu utama.

c. Menu Daftar Kata



Setelah dilakukan penyimpanan kata dalam *database*, selanjutnya pilih menu daftar kata yang terdapat pada menu utama. Dalam menu daftar kata hal yang dapat dilakukan adalah menghapus kata dalam *database.* Tombol

hapus diletakkan pada pojok kiri bawah, sedangkan tombol kembali diletakkan pada pojok kanan bawah. Tombol kembali digunakan untuk kembali ke menu utama.

d. Menu Pencarian Kata

Gambar 4.7 Desain menu pencarian kata

Menu pencarian kata digunakan untuk mencari kata yang terdapat dalam *database*. Tombol cari pada sebelah kiri bawah digunakan untuk proses *searching* dan tombol kembali pada pojok kanan bawah digunakan untuk kembali ke menu utama.

4.3 Penulisan Kode Program

Dalam pembuatan aplikasi Kamus ini, penulis menggunakan bahasa pemrograman J2ME. Berikut dijelaskan fungsi-fungsi *database* yang digunakan pada aplikasi ini:

a. *Method* membuka *record store*

```
openRecordStore ( )
```

Gambar 4.8 Membuka *record store*

Method ini merupakan *method* statik dan digunakan agar *record store* menjadi aktif dan dapat diakses oleh aplikasi.

b. *Method* menghapus *record store*

```
Static void deleteRecordStore (String name )
```

Gambar 4.9 Menghapus *record store*

Method ini digunakan untuk menghapus sebuah *record* yang pernah dibuat sebelumnya.

c. *Method* menghapus *record*

```
Void deleteRecord (int recordID)
```

Gambar 4.10 Menghapus *record*

Method ini digunakan untuk menghapus suatu *record* yang tersimpan dalam suatu *record store.*

Berikut ini adalah beberapa *method* yang digunakan dalam proses penambahan *record* dan dalam memperoleh ID *record.*

d. *Method* menambah dan memperoleh ID dari *record store*

```
// menambahkan record baru
Int addRecord ( )
```

Gambar 3.13 Menambah *record store*

Gambar 4.11 Menambah *record*

```
// memperoleh ID dari record selanjutnya
Int getNextRecordID ( )
```

Gambar 4.12 Memperoleh ID dari *record store*

e. *Method* membaca *record store*

```
Byte getRecord (int recordID)
```

Gambar 4.13 Membaca *record store*

Untuk membaca *record* yang terdapat pada suatu *record store*, dapat digunakan *method* getRecord ( ) dari kelas RecordStore. *Method* tersebut memiliki satu buah parameter, yaitu ID *record.*

f. *Method* mencari *record store*

```
Boolean matches (byte[ ] candidate )
```

Gambar 4.14 Mencari *record store*

Untuk mencari *record* yang telah tersimpan, terdapat *interface* RecordFilter yang digunakan untuk melakukan *filter* terhadap *record-record* yang ada di dalam *record store* sesuai dengan kriteria yang didefinisikan. Dalam *interface* ini, terdapat sebuah *method* yang perlu diimplementasi, yaitu *method* matches ( ); .

g. Eksepsi

Berikut adalah beberapa eksepsi yang terdapat pada paket javax.microedition.rms :

Tabel 1. Daftar Eksepsi dalam RMS

Nama Ekspesi	Keterangan
InvalidRecordException	Dibangkitkan jika mengisikan ID *record* yang salah
RecordStoreException	Dibangkitkan jika terdapat kesalahan umum yang berkaitan dengan *record store*
RecordStoreFullException	Dibangkitkan jika *record store* telah penuh (tidak bisa diisi *record* lagi)
RecordStoreNotFoundException	Dibangkitkan jika *record store* tidak ditemukan
RecordStoreNotOpenException	Dibangkitkan jika akan melakukan operasi terhadap *record store* yang belum dibuka

4.4 Pengujian Program

Aplikasi kamus dalam penelitian ini diuji dengan metode *black box.* Berikut ini adalah detail hasil pengujian tersebut :

4.4.1 Pengujian Menu Tambah Kata

Input 1 : huruf

Output yang seharusnya : huruf

Output yang dihasilkan : huruf

Kesimpulan dari hasil pengujian 1 : Ketika *input* dimasukkan berupa huruf, hasil yang didapatkan setelah diujicoba pun berupa huruf. Maka hasil pengujian dinyatakan valid.

Input 2 : angka

Output yang seharusnya : huruf

Output yang dihasilkan : angka

Kesimpulan dari hasil pengujian 2 : Ketika *input* yang dimasukkan berupa angka, data tetap tersimpan dan hasil yang ditampilkan berupa angka. Seharusnya data yang dihasilkan berupa huruf, maka pengujian dinyatakan tidak valid.

4.4.2 Pengujian Menu Pencarian Kata

Input 1 : 1 huruf

Output yang seharusnya : Berbagai macam kata yang termasuk di dalam huruf awal yang dimasukkan.

Output yang dihasilkan : kata yang termasuk dalam huruf awal.

Kesimpulan dari hasil pengujian 1 : pengujian dinyatakan valid karena data yang dihasilkan sesuai dengan yang diharapkan.

Input 2 : 2 huruf

Output yang seharusnya : Berbagai macam kata yang termasuk di dalam dua huruf awal yang dimasukkan.

Output yang dihasilkan : kata yang termasuk dalam dua huruf awal.

Kesimpulan dari hasil pengujian 1 : pengujian dinyatakan valid karena data yang dihasilkan sesuai dengan yang diharapkan.

Input 3 : 3 huruf

Output yang seharusnya : Berbagai macam kata yang termasuk di dalam tiga huruf awal yang dimasukkan.

Output yang dihasilkan : kata yang termasuk dalam tiga huruf awal.

Kesimpulan dari hasil pengujian 1 : pengujian dinyatakan valid karena data yang dihasilkan sesuai dengan yang diharapkan.

4.5 Hasil

Aplikasi ini dibuat dengan Netbeans 6.9 dengan bahasa pemrograman J2ME. Setelah tahap mengumpulkan data dan informasi dari buku pemrograman, artikel internet, serta hal yang berkaitan dengan penelitian ini, selanjutnya setiap desain tampilan menu serta tombol, diterjemahkan dengan bahasa Java ke dalam bentuk aplikasi yang diinginkan. Dalam

penggunaan aplikasi ini *user* dapat berinteraksi langsung dengan aplikasi kamus menggunakan telepon selular yang telah terinstalasi oleh aplikasi ini.

Proses instalasi dapat dilakukan dengan 2 (dua) tahap, yaitu :

1. *Copy file* JAR/JAD yang dihasilkan oleh kode program dari PC ke dalam telepon selular. Proses ini dapat dilakukan dengan menggunakan *infrared*, *bluetooth*, maupun melalui kabel data. *File* JAR/JAD telah terdapat dalam *folder dist* yang dihasilkan setelah kita mengeksekusi program.
2. Dari telepon selular, pilih *file* JAD/JAR yang baru saja di*copy*. Secara otomatis, AMS (*Application Management Software)* akan memunculkan menu "*install"* pada layer telepon selular. Pilih menu tersebut untuk memulai proses instalasi.

1. Tampilan Aplikasi

Ketika aplikasi pertama kali dijalankan maka menu yang akan tampil adalah sebagai berikut :

a. Menu Utama

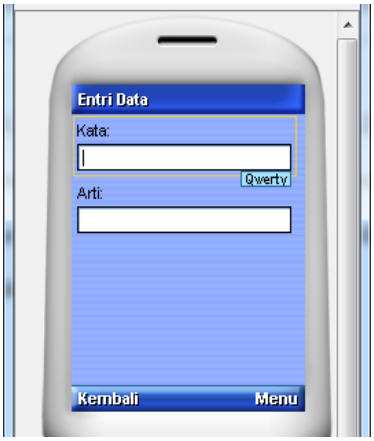
Menu utama aplikasi kamus Bahasa Inggris-Indonesia dapat dilihat seperti di bawah ini :

Gambar 4.15 Menu Aplikasi Kamus Bahasa Inggris-Indonesia

Dalam menu awal terdapat 3 (tiga) menu, menu pertama adalah menu untuk

menambahkan kata, menu kedua adalah untuk melihat daftar kata, dan menu yang ketiga adalah menu untuk mencari kata.

b. Menu Tambah Kata



Gambar 4.16 Menu Tambah Kata

Setelah memilih menu tambah kata, tampilan yang akan dihasilkan adalah seperti gambar di atas. Untuk menambahkan kata ke dalam kamus, maka *user* dapat memilih tombol menu kemudian simpan. Apabila kata berhasil ditambahkan, aplikasi akan menampilkan pesan bahwa kata telah berhasil ditambakan ke dalam *database.*

c. Menu Daftar Kata

Gambar 4.17 Menu Daftar Kata

Di dalam menu daftar kata di atas terdapat tombol kembali dan hapus. Tombol hapus digunakan untuk menghapus daftar kata yang telah dimasukkan ke dalam *database*, sedangkan tombol kembali digunakan jika ingin kembali ke menu awal.

d. Menu Cari Kata

Gambar 4.18 Menu Cari Kata

Dalam menu cari kata, hal yang dapat dilakukan adalah mengetikkan kata yang ingin di cari. Gambar di atas adalah merupakan tampilan dari kamus

Bahasa Indonesia-Inggris. Untuk kamus Bahasa Inggris-Indonesia tampilan yang akan dihasilkan akan sama, hanya yang membedakan adalah bahasa yang digunakan dalam *interface.*

4.6 Pembahasan

1. Karakteristik Telepon selular

Dalam penelitian ini telepon selular yang digunakan sebagai media adalah telepon selular Nokia 6303 dan Samsung B3310 yang tidak bersistem operasi, telepon selular Blackberry 8900 yang bersistem operasi. Telepon selular Nokia 6303 memiliki spesifikasi sebagai berikut :

a. Java MIDP 2.0
b. Memori *internal* 17 MB
c. Slot *MicroSD card up to* 4 GB
d. *Bluetooth* v2.0
e. *Port microUSB*

Untuk telepon selular Blackberry 8900 memiliki spesifikasi sebagai berikut :

a. Java MIDP 2.0
b. Slot *MicroSD card up to* 16 GB
c. Memori *internal* 256 MB
d. *Bluetooth* v2.0

e. *Port microUSB*

f. Blackberry *operating system*

g. Prosesor 512 MHZ

Sedangkan telepon selular Samsung B3310 memiliki spesifikasi sebagai berikut :

a. Java MIDP 2.0

b. Slot *MicroSD*

c. Memori *internal* 43 MB

d. *Bluetooth* v2.0

e. *Port microUSB*

Telepon selular yang digunakan dalam penelitian ini ada yang tidak memiliki sistem operasi dan ada yang memiliki sistem operasi, tetapi keduanya mendukung Java MIDP 2.0 yang diperlukan dalam aplikasi kamus ini. Aplikasi kamus ini dapat diinstal dalam memori telepon selular maupun memori *external*. Hal ini tidak mempengaruhi kinerja telepon selular. Aplikasi ini cukup membebani memori, tetapi dalam pengisian kata hanya terbatas sebanyak 32 karakter.

2. **Perbandingan Kinerja Aplikasi**

Setelah dilakukan pemindahan *file* JAD/JAR dari PC ke masing-masing telepon dengan spesifikasi di atas, hasil perbandingan yang diperoleh adalah

sebagai berikut :

a. Telepon selular Nokia 6303 yang Tidak Bersistem Operasi

Pada saat data dipindahkan ke dalam perangkat, telepon selular jenis ini tidak memerlukan proses instalasi, melainkan hanya langsung membuka aplikasi JAD/JAR yang telah dipindahkan dari PC. Pada telepon selular ini memori yang diperlukan untuk aplikasi ini adalah 16 kb, dan setelah diujicoba dengan memasukkan beberapa kata dalam Bahasa Inggris/Indonesia memori berubah. Selanjutnya aplikasi diuji coba untuk melakukan kegiatan lain seperti menerima telepon, atau mendengarkan musik (*multitasking*) saat aplikasi aktif. Dalam hal ini telepon selular dapat melakukan 2 (dua) kegiatan tersebut pada saat aplikasi kamus sedang berjalan. Untuk kecepatan penyimpanan, penghapusan, serta pencarian dalam telepon selular ini dinilai relatif cepat.

b. Telepon selular Blackberry 8900 yang Bersistem Operasi

Berbeda dengan telepon selular sebelumnya, telepon selular jenis ini memerlukan proses instalasi yang kurang lebih membutuhkan waktu sekitar 30 detik. Setelah proses instalasi berlangsung, aplikasi akan otomatis berpindah ke dalam memori telepon selular. Memori yang dibutuhkan setelah proses instalasi berlangsung adalah 6936 *bytes*. Memori tidak berubah setelah di uji coba dengan memasukkan beberapa kata. Uji coba selanjutnya adalah apakah telepon selular ini *multitasking* atau tidak. Setelah diujicoba, telepon selular ini dapat melakukan kegiatan secara bersamaan ketika aplikasi kamus dibuka. Untuk proses pencarian, penyimpanan, serta penghapusan kata dalam

telepon selular ini relatif cepat.

c. Telepon selular Samsung B3310 yang Tidak Bersistem Operasi

Berbeda dengan Nokia 6303, telepon selular ini memerlukan proses instalasi untuk membuka aplikasi kamus ini. Waktu yang dibutuhkan kira-kira 30 detik, proses instalasi hampir sama dengan Blackberry 8900 yang memiliki sistem operasi. Setelah proses instalasi berlangsung, maka aplikasi secara otomatis akan berpindah ke dalam memori telepon selular. Memori yang dibutuhkan setelah proses instalasi adalah 6 Kb. Untuk proses penambahan, penghapusan, serta pencarian kata relatif cepat. Sedangkan ketika diujicoba *multitasking*, telepon selular dapat melakukan beberapa kegiatan secara bersamaan, seperti mendengar musik dan menerima telepon.

Tabel 2. Perbandingan antara Nokia, Blackberry, dan Samsung

	Nokia	Blackberry	Samsung
Proses Penginstalan	Tidak diperlukan	Diperlukan	Diperlukan
Lama instalasi	-	30 detik	30 detik
Memori	16 kb	6936 *bytes*	6 kb

3. Kelebihan dan Kekurangan Aplikasi

Dalam proses pembuatan aplikasi kamus Bahasa Inggris-Indonesia ini terdapat beberapa kelebihan dan kekurangannya, antara lain adalah :

a. Kelebihan Aplikasi

Beberapa kelebihan dari aplikasi kamus ini adalah :

1. Tidak membebani memori ketika melakukan pencarian kata.
2. Memori yang digunakan tidak terlalu besar.
3. *User friendly.*

b. Kekurangan Aplikasi

Beberapa kekurangan dari aplikasi kamus ini adalah sebagai berikut :

1. Karakter yang disimpan terbatas hanya sebanyak 32 karakter saja.
2. Harus diinstal pada telepon selular yang mendukung Java MIDP 2.0.

BAB V

KESIMPULAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan dapat disimpulkan sebagai berikut:

1. Aplikasi kamus Bahasa Inggris-Indonesia ini cukup membebani kinerja telepon selular, baik itu telepon selular yang bersistem operasi, maupun telepon selular yang tidak bersistem operasi.
2. Pada telepon selular jenis Nokia aplikasi ini tidak harus diinstal dalam memori telepon selular, dapat pula diinstal pada memori tambahan, sedangkan pada telepon selular jenis Blackberry dan Samsung aplikasi akan otomatis terinstal ke dalam memori telepon.
3. Data yang dimasukkan masih bersifat acak atau tidak berurutan, sehingga digunakan metode pencarian data untuk mempermudah dalam pencarian kata.

4. Kata yang dimasukkan terbatas jumlahnya, pembatasan dilakukan pada saat menambahkan kata. Jumlah karakter yang ditetapkan adalah 32 karakter.

5.2 Saran

1. Aplikasi ini masih memerlukan pengembangan yang lebih baik lagi terutama penggunaan algoritma pencocokan kata yang lebih bervariasi.
2. Aplikasi ini belum diujicobakan pada perangkat PDA, Ipad dan perangkat *mobile* lainnya. Ujicoba penggunaan aplikasi ini terhadap perangkat-perangkat tersebut masih perlu dieksplorasi, terutama jika kompatibilitasnya rendah.
3. Proses pengurutan kata dalam aplikasi ini tidak dilakukan karena ID kata diberikan oleh aplikasi saat kata di*entry*. Oleh karena itu, *coding* aplikasi masih dapat diperbaiki agar kata-kata yang tersimpan telah terurut secara alfabet sehingga mempercepat proses pencarian kata.

.

DAFTAR PUSTAKA

Sun Microsystems Inc., *Core 2 J2ME Technology and MIDP*, Sun Microsystems Press and Prentice-Hall, 2002.

Johannes. *Java ME*. Jasakom, 2010.

Raharjo, Budi., Imam Heryanto., Arif Haryono. 2010. *Tuntunan Pemrograman Java untuk Handphone*. Informatika. Bandung.

Pressman, Roger S,. 1997. *Rekayasa Perangkat Lunak Pendekatan Praktisi buku 1*, Andi Offset, Yogyakarta.

Hakim, Rachmad,. Sutarto. *Mastering Java*. Elex Media Komputindo. Jakarta, 2009.

JENI (https://sites.google.com/a/meruvian.org/meruvian/jeni) diakses tanggal 10 Mei 2010 pukul 14.30

http://www.teknologinet.com/2009/01/spesifikasi-dan-harga-blackberry-javelin-8900.html diakses tanggal 01 Oktober 2010 pukul 19.20

http://lifestyle.iloveindia.com/lounge/blackberry-curve-8900-review-4257.html diakses tanggal 01 Oktober 2010 pukul 19.25

http://bloggerceria.com/2010/01/samsung-b3310-muncul-spesifikasi-harga.html diakses tanggal 17 Oktober 2010 pukul 13.30

Khoiriyah, Atik,. 2010. website : http://atikkhoiriyah.wordpress.com/2010/06/15/j2me-midp-record-store/ diakses tanggal 20 Oktober 2010 pukul 19.30

student.eepis-its.edu/~abdan/Kuliah/J2ME/UTS_J2ME.doc diakses tanggal 20 Oktober 2010 pukul 19.35

LAMPIRAN

CODING

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.rms.*;

public class KamusIndonesia extends MIDlet
  implements CommandListener {

  private Display display;
  private Form form;
  private RecordStore rs;
  private RecordEnumeration re;
  private ChoiceGroup choicegroup;
  private Alert alert;
  private List list;

  //untuk proses entri data
  private Form entri;
  private TextField tfinput, tfarti;

  //proses cari data
  private TextField tfCari;
  private Form temukan;

  private final Command cmdKeluar =
    new Command("Keluar", Command.EXIT, 7);
  private final Command cmdPilih =
```

```
  new Command("Pilih", Command.OK, 4);
private final Command cmdSimpan =
  new Command("Simpan", Command.SCREEN, 1);
private final Command cmdHapus =
  new Command("Hapus", Command.SCREEN, 1);
private final Command cmdKembali =
  new Command("Kembali", Command.BACK, 2);
private final Command cmdPencarian =
  new Command ("Pencarian",Command.SCREEN, 1);
private final Command cmdCari =
  new Command ("Cari",Command.OK, 4);

public KamusIndonesia() {
  display = Display.getDisplay (this);

  alert = new Alert(null);
  alert.setTimeout(Alert.FOREVER);

  list = new List(null, Choice.IMPLICIT);

  rs = null;

  // membuat atau membuka record store
  try {
    rs = RecordStore.openRecordStore("contohDB", true);
  } catch (RecordStoreException rse) {
    alert.setString("Daftar Kata tidak dapat dibuka. " +
                    "Aplikasi akan dihetikan");
    alert.setType(AlertType.ERROR);
    display.setCurrent(alert, null);
    System.exit(1);
  }
}

public void startApp() {
  form = new Form("Indonesia-Inggris");

  choicegroup = new ChoiceGroup("Menu:", Choice.EXCLUSIVE);
  choicegroup.append("Tambah Kata", null);
  choicegroup.append("Daftar Kata", null);
  choicegroup.append("Cari Kata", null);

  form.append(choicegroup);
  form.addCommand(cmdKeluar);
  form.addCommand(cmdPilih);
  form.setCommandListener(this);
```

```
 display.setCurrent(form);
 }


public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable s) {
 if (c == cmdKeluar) {
  destroyApp(false);
  notifyDestroyed();

 } else if (c == cmdPilih) {
  switch (choicegroup.getSelectedIndex()) {
   case 0: {
    entriData();
    break;
   }
   case 1: {
    lihatRecord();
    break;
   }
   case 2:{
    cariKata ();
    break;
    }
  }
 } else if (c == cmdKembali) {
  display.setCurrent(form);
 } else if (c == cmdSimpan) {
  alert.setType(AlertType.INFO);
  if (!tfinput.equals("") && !tfarti.equals("")) {
   tambahRecord(tfinput.getString(), tfarti.getString());
   alert.setString("Data baru telah berhasil disimpan");
   display.setCurrent(alert, form);
  } else {
   alert.setString("Data Kata dan Arti " +
                   "tidak boleh kosong");
   display.setCurrent(alert, entri);
  }
 } else if (c == cmdHapus) {
  int pos =
   list.getString(list.getSelectedIndex()).indexOf(" [");
```

```
    String input = list.getString(
     list.getSelectedIndex()).substring(0, pos);
     hapusRecord(input);
  }
   else if (c == cmdCari) {
       display.setCurrent (list);
       tampilkanData (true);
   }
}


public void tambahRecord(String input, String arti) {
  byte[] temp = null;
  try {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    DataOutputStream dos = new DataOutputStream(baos);
    dos.writeUTF(input);
    dos.writeUTF(arti);
    temp = baos.toByteArray();
  } catch (IOException ioe) {
    ioe.printStackTrace();
  }
  try {
    rs.addRecord(temp, 0, temp.length);
  } catch (RecordStoreNotOpenException rsnoe) {
    rsnoe.printStackTrace();
  } catch (RecordStoreException rse) {
    rse.printStackTrace();
  }
}

public void lihatRecord() {
  byte[] temp = null;
  list.setTitle("Daftar Kata");
  list.deleteAll();
  try {
    re = rs.enumerateRecords(null, null, false);
    while (re.hasNextElement()) {
      int i = re.nextRecordId();
      temp = rs.getRecord(i);
      ByteArrayInputStream bais =
        new ByteArrayInputStream(temp);
      DataInputStream dis = new DataInputStream(bais);
      try {
        String input = dis.readUTF();
        String arti = dis.readUTF();
```

```
        list.append(input + " [" + arti + "]", null);
      } catch (IOException ioe) {
        ioe.printStackTrace();
      }
    }

    list.addCommand(cmdKembali);
    list.addCommand(cmdHapus);
    list.setCommandListener(this);
    display.setCurrent(list);
  } catch (InvalidRecordIDException invID) {
    invID.printStackTrace();
  } catch (RecordStoreNotOpenException rsnoe) {
    rsnoe.printStackTrace();
  } catch (RecordStoreException rse) {
    rse.printStackTrace();
  }

}

public void hapusRecord(String input) {
  byte[] temp = null;
  try {
    re = rs.enumerateRecords(null, null, false);
    while (re.hasNextElement()) {
      int i = re.nextRecordId();
      temp = rs.getRecord(i);
      ByteArrayInputStream bais =
        new ByteArrayInputStream(temp);
      DataInputStream dis = new DataInputStream(bais);
      try {
        String vinput = dis.readUTF();
        if (vinput.equals(input)) {
          rs.deleteRecord(i);
          break;
        }
      } catch (IOException ioe) {
        ioe.printStackTrace();
      }
    }
    re.rebuild();
    lihatRecord();
  } catch (RecordStoreNotOpenException rsnoe) {
    rsnoe.printStackTrace();
  } catch (RecordStoreException rse) {
    rse.printStackTrace();
```

```
    }
  }

  public Form entriData() {
    entri = new Form("Entri Data");
    tfinput = new TextField("Kata:", null, 15, TextField.ANY);
    tfarti = new TextField("Arti:", null, 15,
                           TextField.ANY);
    entri.append(tfinput);
    entri.append(tfarti);
    entri.addCommand(cmdSimpan);
    entri.addCommand(cmdKembali);
    entri.setCommandListener(this);
    display.setCurrent(entri);
    return entri;
  }
public Form cariKata ()  {
    temukan = new Form("Pencarian Kata");
    tfCari = new TextField("Masukkan Kata:", null, 15, TextField.ANY);

    temukan.append(tfCari);
    temukan.addCommand(cmdCari);
    temukan.addCommand(cmdKembali);
    temukan.setCommandListener(this);
    display.setCurrent(temukan);
    return temukan;
    }


  public void tampilkanData(boolean filter) {
    byte[] temp = null;
    RecordEnumeration re = null;

    list.deleteAll();
    try {
      if (filter) {
        String s = tfCari.getString().toLowerCase();
        CariNama cn = new CariNama(s);
        re = rs.enumerateRecords(cn, null, false);
        list.setTitle("Hasil Pencarian");
        list.addCommand(cmdKembali);
        list.setCommandListener(this);

      }
      while (re.hasNextElement()) {
        int i = re.nextRecordId();
```

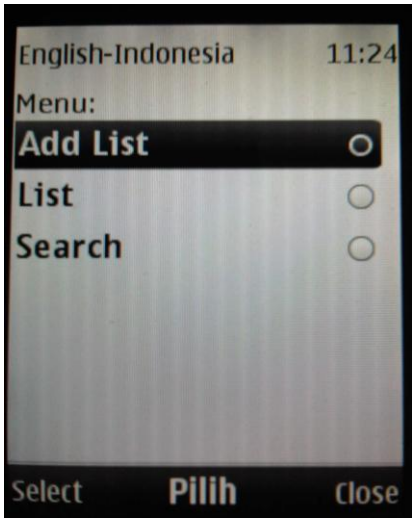
```
        temp = rs.getRecord(i);
        ByteArrayInputStream bais =
          new ByteArrayInputStream(temp);
        DataInputStream dis = new DataInputStream(bais);
        try {
          String input = dis.readUTF();
          String arti  = dis.readUTF();
          list.append(input + " [" + arti + "]", null);
        } catch (IOException ioe) {
          ioe.printStackTrace();
        }
      }
      display.setCurrent(list);
    } catch (InvalidRecordIDException invID) {
      invID.printStackTrace();
    } catch (RecordStoreNotOpenException rsnoe) {
      rsnoe.printStackTrace();
    } catch (RecordStoreException rse) {
      rse.printStackTrace();
    }

    display.setCurrent(list);
  }
 // inner class untuk mencari data
  class CariNama implements RecordFilter {

    private String karakterAwal;

    public CariNama(String karakterAwal) {
      this.karakterAwal = karakterAwal.toLowerCase();
    }

    public boolean matches(byte[] candidate) {
      String s =
        new String(candidate).toLowerCase().substring(2);
      return s.startsWith(karakterAwal);
    }
  }
}
```

GAMBAR

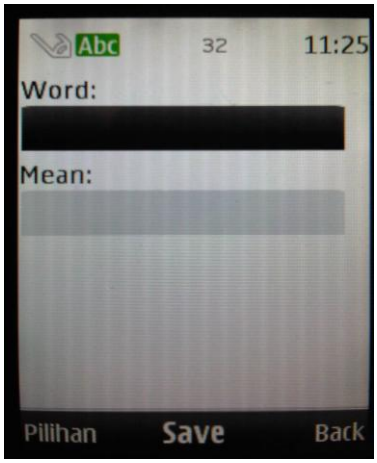
1. Nokia 6303

a. Instalasi

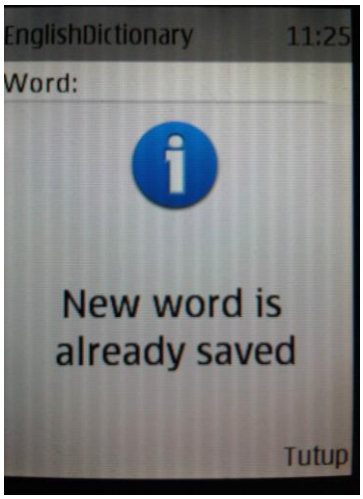
b. Menu Utama



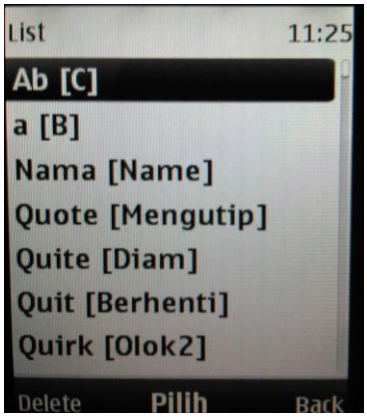
c. Menu tambah kata



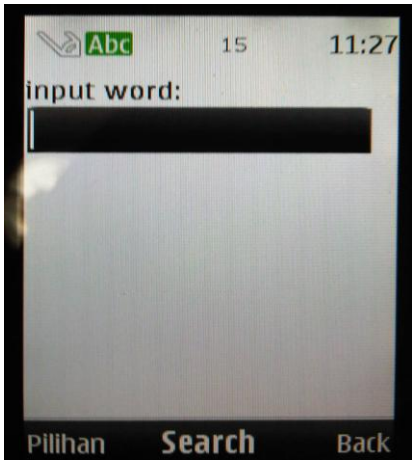
d. *Alert* jika kata berhasil disimpan



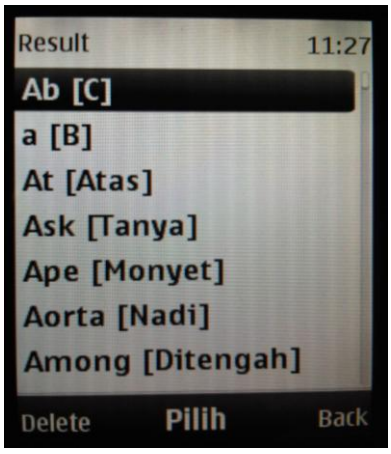
e. Menu Daftar Kata



f. Menu Cari Kata



g. Menu Hasil Pencarian



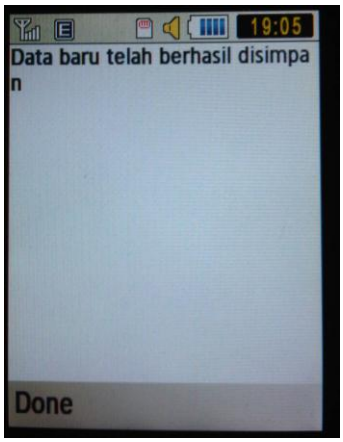
2. Samsung

a. Instalasi

b. Menu Utama

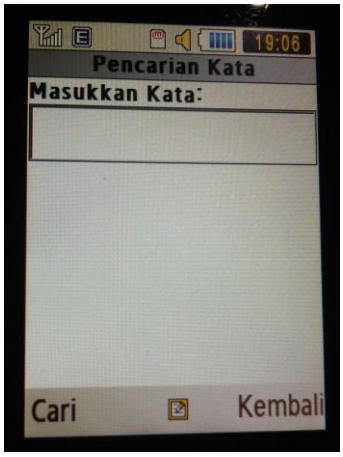
c. Menu Tambah Kata

d. *Alert* Jika pesan berhasil disimpan



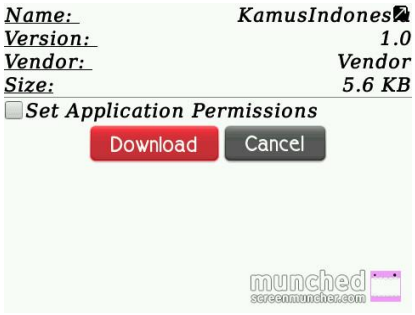
e. Menu Daftar Kata

f. Menu Cari Kata



3. BlackBerry

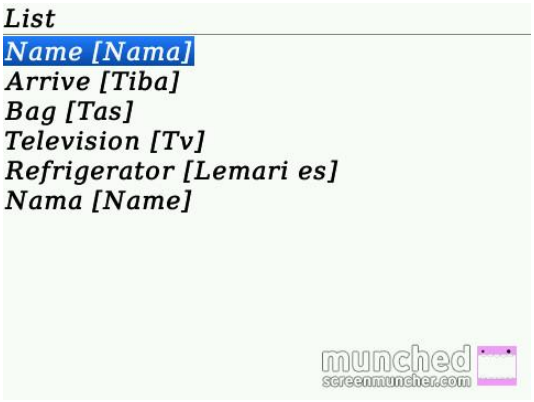
a. Menu Instalasi



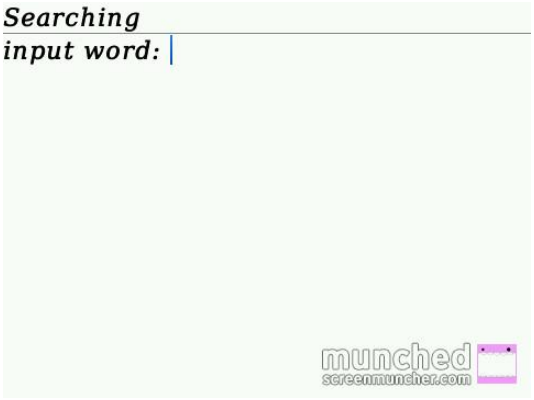
b. Menu Utama

c. Menu Tambah Kata

d. Daftar Kata



e. Menu Pencarian Kata



f. Hasil Pencarian