BAB II
LANDASAN TEORI

II.1 Pengantar

Pada bab ini akan dijelaskan mengenai beberapa teori yang ada pada literatur yang dijadikan acuan untuk pembangunan sistem, diantaranya meliputi layanan-layanan *social networking*, SMS (*Short Message Service*), HTTP (*Hypertext Transfer Protocol*), teknologi GPRS (*General Packet Radio Service*), teknologi Java dan J2ME (*Java 2 Micro Edition*).

II.2 *Social Networking* dan *Mobile Social Networking*

Layanan *social networking* dibangun untuk berbagai macam komunitas agar bisa saling bertukar pikiran dan informasi mengenai hobi, afiliasi, dan aktivitas-aktivitas para anggotanya. Sebagian besar layanan tersebut berbasis *web* dan menyediakan berbagai macam cara interaksi antar pemakainya, misalnya *chat, messaging, email* dan *mailing list, video, voice chat, file sharing, blogging, discussion groups,* dan sebagainya. Beberapa layanan *social networking* memiliki fitur-fitur tambahan, seperti kemampuan untuk membuat grup-grup yang berbagi *interests* dan afiliasi yang sama, *upload video,* dan media diskusi dalam suatu forum.

Mobile Social Networking adalah *social networking* dimana dua atau lebih individu yang memiliki kesamaan *interests,* saling berbagi, bertukar pikiran dengan menggunakan perangkat *mobile.* Seperti *social networking* berbasis *web, mobile social networking*

terjadi dalam komunitas-komunitas *virtual*. Perkembangan teknologi *mobile* saat ini sudah sangat maju sehingga perangkat-perangkat *mobile* bisa memiliki kemampuan seperti *desktop* komputer untuk mengakses jaringan internet. Hal ini membuat layanan-layanan *social networking* berbasis *web*, seperti *Instant Messaging (Mobile Chat)*, *Internet Forums*, dan *Mailing List* bisa diaplikasikan ke dalam perangkat *mobile*. Distribusi antar komunitas dilakukan dengan menggunakan *start page* pada *browser* perangkat *mobile*, misalnya yang sudah banyak dikenal seperti *JuiceCaster*, *Jumbuck*, dan *AirG*. Sedangkan beberapa layanan *social networking* dalam perangkat *mobile* yang menggunakan metode lain juga sudah banyak dikembangkan seperti *Mobimii, MocoSpace, dan BluePulse*. Ini dipicu oleh perkembangan teknologi *wireless network* seperti SMS, WAP, GPRS, 3G, *Java*, BREW, *Symbian, Windows Mobile, dan i-mode*. (en.wikipedia.org/wiki/Social network service)

II.2.1 *Instant Messaging* (IM) / *Chatting*

Instant Messaging (IM) adalah suatu bentuk komunikasi *real time* antara dua atau lebih orang yang berbasis teks melalui jaringan internet. Dalam *Instant Messaging* pihak-pihak yang berkomunikasi sama-sama *online*, sehingga komunikasi bisa dilakukan secara simultan. Beberapa sistem IM memperbolehkan para pemakainya menggunakan *webcam* dan *Microphone* sehingga dengan fitur ini para pemakai bisa melakukan percakapan secara *real time*. Fitur tambahan lain yang dimiliki IM misalnya *group chatting*, layanan *conference*, dan transfer *file*. Saat ini terdapat banyak *chat room*

ataupun *messaging clients* mulai dari yang hanya berbasis teks maupun berbasis GUI (*Graphical User Interface*) seperti Internet *Relay Chat* (IRC), ICQ, AOL *Instant Messenger*, Yahoo! Messenger, dan sebagainya. (en.wikipedia.org/wiki/instant messaging)

II.2.2 *Internet Forum*

Internet Forum adalah aplikasi *web* untuk menangani diskusi-diskusi tentang suatu topik dan *posting user-generated content* dalam suatu komunitas *social network*. *Internet Forum* juga biasanya mengacu pada *web forums, message boards, discussion boards, discussion groups, discussion forums, bulletin boards, fora*, atau hanya *forums* saja. Pesan-pesan dalam forum ditampilkan secara kronologikal atau sebagai *threaded discussions*. Dalam *forum*, para partisipan tidak perlu *online* secara simultan untuk mengirim atau menerima pesan-pesan, hal inilah yang membedakan *forum* dengan *IM*. (en.wikipedia.org/wiki/Internet forum)

Terdapat beberapa jenis *user* dalam suatu *forum* diantaranya yaitu anggota biasa, moderator, dan *administrator*. Anggota suatu *forum* yang diidentifikasi berdasarkan *username* yang unik, selain dapat me-*posting* juga bisa mendapatkan hak tambahan misalnya kemampuan untuk mengedit *posting*-an sebelumnya, memulai suatu topik baru, dan mengontrol *settings* dan *profiles* individu mereka. Seorang *administrator* suatu *forum* memiliki hak dan kemampuan untuk meng-edit, menghapus, memindahkan atau memodifikasi *thread* dalam *forum*. *Administrator* biasanya juga memiliki kemampuan untuk menutup suatu *board,* merubah *item-item software* secara

global, memodifikasi *board,* dan menghapus atau membuat *user* baru. Sedangkan *moderator* memiliki hak-hak dibawah *administrator,* misalnya meng-*edit,* menghapus, dan memindahkan *thread,* dan memberikan peringatan, ini memungkinkan hak moderator bisa diberikan kepada anggota forum.

II.2.3 *Mailing List*

Mailing list adalah kegunaan khusus dari *e-mail* yang memperbolehkan untuk mendistribusikan informasi secara luas kepada banyak pengguna internet. Saat ini *mailing list* merupakan layanan yang cukup laris yang digunakan komunitas-komunitas *social network*. *Mailing list* memiliki beberapa elemen yaitu daftar alamat *e-mail*, orang-orang yang menerima pesan melalui alamat-alamat *e-mail* tersebut, pesan *e-mail* yang dikirimkan ke alamat-alamat *e-mail* tersebut, dan sebuah *reflector*, yang merupakan alamat *e-mail* tunggal, yang didesain sebagai *recipient* dari pesan *e-mail* tersebut, dan kemudian mengirimkan pesan tersebut ke semua *subscribers*. *Mailing list* diotomatisasi dengan menggunakan *software mailing list* khusus dan sebuah alamat *reflector* khusus yang di *setup* pada *server* yang memiliki kemampuan tersebut. (en.wikipedia.org/wiki/Mailing list)

II.3 *Short Message Service* (SMS)

SMS adalah protokol komunikasi yang digunakan untuk pertukaran pesan-pesan pendek (maks 160 karakter) antar perangkat *mobile*. Mulanya SMS dibangun sebagai bagian dari seri standar GSM, namun saat ini sudah

terintegrasi dengan standar *mobile* yang lainnya seperti *ANSI CDMA Network* dan *Digital AMPS*, yang merupakan jaringan satelit dan *landline*. Teknologi SMS telah memfasilitasi perkembangan *text messaging*. Saat ini teknologi text messaging berbasis SMS sudah sangat dikenal dikalangan masyarakat.

SMS *Gateway* merupakan layanan yang menyediakan antarmuka antara layanan SMS dengan protokol dan layanan lain, misalnya untuk menghubungkan dengan layanan *Instant Messaging* (IM), *World Wide Web* (WWW), komputer *desktop*, atau *landline* telepon. Penyedia layanan SMS *gateway* memfasilitasi lalu lintas SMS antara pelaku bisnis dan *mobile subscribers*. (en.wikipedia.org/wiki/Short message service)

Berikut ini adalah beberapa layanan *social networking* yang menggunakan teknologi SMS :

1. *SMS chat*

   Dengan SMS *chat*, seluruh SMS yang dikirim maupun diterima dari seseorang akan muncul saling bersahutan seperti layaknya melakukan *chatting* dengan *Instant Messenger*. Sekarang sudah cukup banyak beredar aplikasi-aplikasi untuk SMS chat, salah satunya yaitu *SMS Chat Symbianwave* dan *SMS Chat Windows Mobile*.

2. *SMS Group*

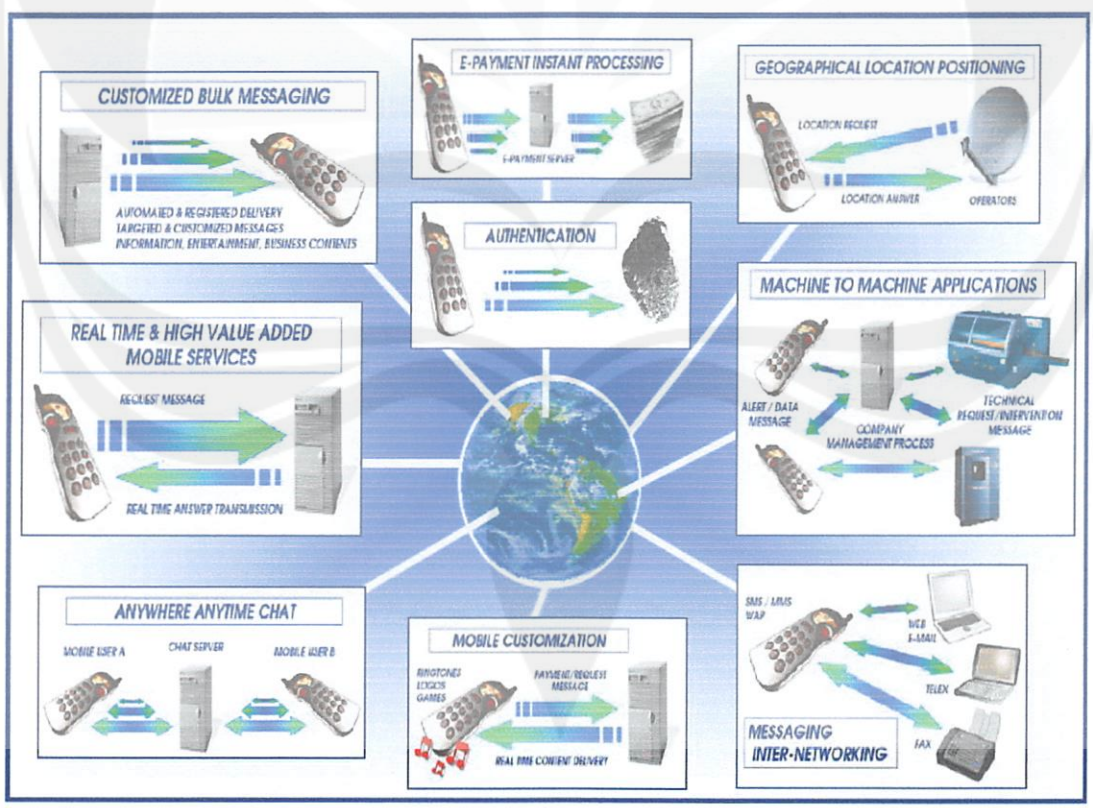
   SMS group atau biasa disebut *SMS premium* merupakan layanan *social networking* yang menyediakan distribusi *content* ataupun hanya *chatting* seperti pada *SMS chat* kepada para *member*-nya. Fitur-fitur dalam *SMS Group* contohnya REG (*subscribe*) dan UNREG

(*unsubscribe*). Implementasi SMS Group saat ini misalnya untuk *TV voting, content delivery , entertainment,* dan *chatting* antar *member*-nya.

3. *SMS Broadcast*

SMS broadcast merupakan layanan yang memanfaatkan *SMS gateway*. Layanan ini memungkinkan pengiriman pesan SMS kepada banyak orang (*one to many*) dan hanya bisa terjadi satu arah. Biasanya layanan *SMS broadcast* dipergunakan untuk media promosi perusahaan.

Masih banyak layanan-layanan yang dikembangkan dengan menggunakan teknologi SMS, seperti yang dijelaskan oleh gambar berikut ini :



Gambar 2.1 Layanan-layanan yang menggunakan SMS (Mavrakis,2004)

II.4 *General Packet Radio Service* (GPRS)

Teknologi transmisi data GSM berupa GPRS (*General Packet Radio Service*) adalah sebuah teknologi yang dipergunakan untuk pelayanan data *wireless* seperti pada *wireless* internet atau intranet serta pelayanan *multimedia*. Juga biasanya disebut sebagai GSM-IP (*Internet Protocol*), karena akan menghubungkan pengguna dengan ISP (*Internet Service Provider*).

Secara teoritis kecepatan maksimum sistem ini 171.2 kbps. Ini adalah 3 kali lebih cepat dari transmisi data dari sistem *network* telekomunikasi *fixed* yang ada dan 10 kali lebih cepat dari *Circuit Switched Data* pada sistem *network* telekomunikasi *wireless* GSM saat ini. Dengan kecepatan pengiriman data, maka sistem ini tentu saja akan memberikan pelayanan yang lebih murah dibanding dengan SMS atau *Circuit Switched Data*.

GPRS adalah sebuah sistem yang terhubung terus, dimana informasi dapat segera dikirim atau diterima saat diperlukan. Tidak perlu adanya *dial-up modem*. GPRS memberikan fasilitas baru untuk beberapa aplikasi yang sebelumnya tidak ada pada *network* GSM, karena adanya keterbatasan pada kecepatan di *Circuit Switched Data* (9.6 kbps) dan panjang *message*/berita pada *Short Message Service* (160 *characters*). GPRS akan dapat secara penuh mengaplikasikan internet seperti yang biasa kita dapatkan di *desktop* komputer, dari *web browsing* sampai dengan *chat*. Selain itu GPRS akan memberikan kemudahan transfer *file*, dan *home automation*. (sumpahpalapa.com/wap/gprs.php)

II.5 *Hypertext Transfer Protocol* (HTTP)

Hypertext Transfer Protocol (HTTP) adalah protokol komunikasi untuk transfer informasi dalam intranet dan *World Wide Web*. HTTP secara umum ditujukan untuk menyediakan cara untuk mem-*publish* dan menerima halaman-halaman *hypertext*, misalnya HTML (*Hypertext Markup Language*) melalui internet. (en.wikipedia.org)

Perkembangan HTTP dikoordinasi oleh *World Wide Web Consortium* (W3C) dan *Internet Engineering Task Force* (IETF), dan mencapai puncak publikasi pada saat dijadikan salah satu seri *Request For Comments* (RFCs), yaitu RFC 2616 pada Juni 1999, yang mana didefinisikan versi HTTP/1.1 yang saat ini banyak digunakan *web browser*. Sedangkan versi sebelumnya, yaitu HTTP/0.9 dan HTTP/1.0 didefinisikan pada RFC 1945. (McClure, 2003)

HTTP adalah sebuah metode *request/response* standar antara *client* dan *server*. Mula-mula *client* mengirimkan HTTP *request*, menggunakan berbagai macam metode dan *request headers*, sesuai dengan jenis permintaan, lalu *server* mengirimkan kode HTTP *response*, menggunakan berbagai macam *request headers*, untuk melayani permintaan *client*. (Mansfield, 2004)

II.5.1 HTTP *Request*

Dibandingkan pendahulunya, HTTP/1.1 secara signifikan meningkatkan dukungan terhadap metode-metode HTTP *request*. Berikut ini adalah metode-metode HTTP *request* yang didukung HTTP/1.1, yaitu :

1. CONNECT, metode yang digunakan dengan sebuah *proxy* yang memiliki kemampuan untuk bertukaran secara dinamis dengan mode SSL *tunneling*.

2. DELETE, metode yang digunakan untuk meminta *server* asli menghapus sumber yang disebutkan.
3. GET, metode yang digunakan untuk mengambil informasi yang diminta dari sistem *file*.
4. HEAD, metode yang digunakan untuk mengembalikan informasi *meta (header)* pada data yang diminta.
5. OPTIONS, metode yang digunakan untuk meminta informasi tentang pilihan-pilihan komunikasi yang tersedia pada sumber yang diminta.
6. POST, metode yang digunakan untuk menerima informasi yang diikutsertakan dan menindaklanjutinya.
7. PUT, metode yang digunakan untuk meminta elemen-elemen yang diikutsertakan disimpan dalam sumber-sumber yang disediakan.
8. TRACE, metode yang digunakan untuk mengirimkan sebuah *request* untuk pesan *loopback*.

II.5.2 HTTP *Response*

Request dari *client* ditangani oleh *server* dan diresponnya secara tepat. Dalam respon tersebut, *server* mengirimkan kembali serangkaian komponen pesan yang dapat dikategorikan menjadi :

1. Kode Respon, sebuah kode angka yang sesuai dengan respon yang bersangkutan, misalnya :
   - 1.a 200 OK, artinya *request* berhasil dijalankan.
   - 1.b 400 *Bad Request*, artinya *request* tidak dimengerti *server*.
   - 1.c 403 *Forbidden*, artinya *server* memahami *request* tetapi menolak untuk merespon.

1.d 404 *Not Found*, artinya sumber-sumber yang diminta tak ditemukan.

2. *Field Header*, adalah informasi tambahan mengenai respon tersebut.
3. Data, merupakan isi atau badan (*body*) dari respon.

II.6 *Java*

Java adalah bahasa pemrograman yang dikembangkan oleh Sun Microsystems. Alasan utama pembentukan bahasa *Java* adalah untuk membuat aplikasi-aplikasi yang dapat diletakan diberbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control*, sehingga *Java* harus bersifat portabel. (Raharjo, 2007)

II.6.1 Kelebihan Java

Terdapat beberapa kelebihan bahasa pemrograman *Java* yaitu :

1. Berorientasi Objek

   Java merupakan bahasa pemrograman berorientasi objek. Ini berarti bahwa dalam sistem berorientasi objek sebuah *class* merupakan *instance* dari sebuah *object* yang berisi data dan *methods* untuk menggambarkan status dan perilaku dari *object* tersebut. Dengan paradigma ini para programer *Java* hanya fokus pada data dan *methods* yang memanipulasi data tersebut dalam aplikasi.
2. Terinterpretasi dan Portabel

   Java adalah sebuah bahasa yang terinterpretasi (Flanagan,1997). *Java compiler* tidak men-*generate* kode mesin tetapi *byte-codes* dan untuk

menjalankan sebuah program *Java* (*run-time*) maka dibutuhkan *Java interpreter*, yaitu *Java Virtual Machine* (JVM). Karena *Java byte-codes* adalah *platform* yang independen, maka program-program *Java* dapat dijalankan di bermacam-macam *platform* yang memiliki JVM dan dapat didistribusikan secara bebas. Hal inilah yang memicu jargon *Java* yaitu "*write once, run anywhere* ™".

3. Dinamis dan Terdistribusi

   Java adalah bahasa pemrograman yang dinamis. Semua *class Java* dapat di-*load* menjadi *Java interpreter* setiap saat. *Java* juga disebut sebagai bahasa pemrograman yang terdistribusi. Ini berarti *Java* menyediakan dukungan yang sangat baik untuk pemrograman jaringan dan internet.

4. Sederhana dan Andal

   Java adalah bahasa pemrograman yang mudah dipelajari karena memiliki jumlah konstruksi bahasa yang relatif kecil. *Java* juga didesain untuk pembangunan perangkat lunak yang andal. *Java* tidak mengeliminasi kebutuhan terhadap *Software Quality Assurance* (SQA).

5. *Secure*

   Keamanan adalah salah satu aspek penting yang diusung bahasa *Java* karena sifat bahasa *Java* yang terdistribusi dan independen. Tanpa jaminan keamanan, setiap orang tidak akan men-*download* kode-kode *Java* dari sembarang situs di internet dan menjalankannya di komputer pribadi. Untuk itu *Java* menyediakan beberapa kontrol lapisan

keamanan yang memberikan perlindungan terhadap kode-kode berbahaya.

6. *High Performance*

   Sebagai bahasa yang memberikan dukungan terhadap pemrograman aplikasi berbasis GUI (*Graphical User Interface*) dan jaringan, *Java* memiliki performansi yang sangat baik.

7. *Multithreaded*

   Java memberikan dukungan untuk eksekusi *multiple thread* yang dapat menangani *tasks* yang berbeda-beda. Dukungan terhadap *multithreading* ini membuktikan performansi yang interaktif dari *graphical application* untuk *user*. (Flanagan, 1997)

8. *Garbage Collection*

   Garbage collection merupakan teknik manajemen *memory* yang digunakan program *Java* untuk menangani alokasi *memory* dinamis. Dengan teknik ini para *programmer* tidak akan mengalokasikan memory secara manual karena alokasi memory secara otomatis akan dilakukan oleh *Java Runtime Environment*.

II.6.2 Arsitektur *Java*

Arsitektur bahasa *Java* terdiri dari :

1. *Compiler*

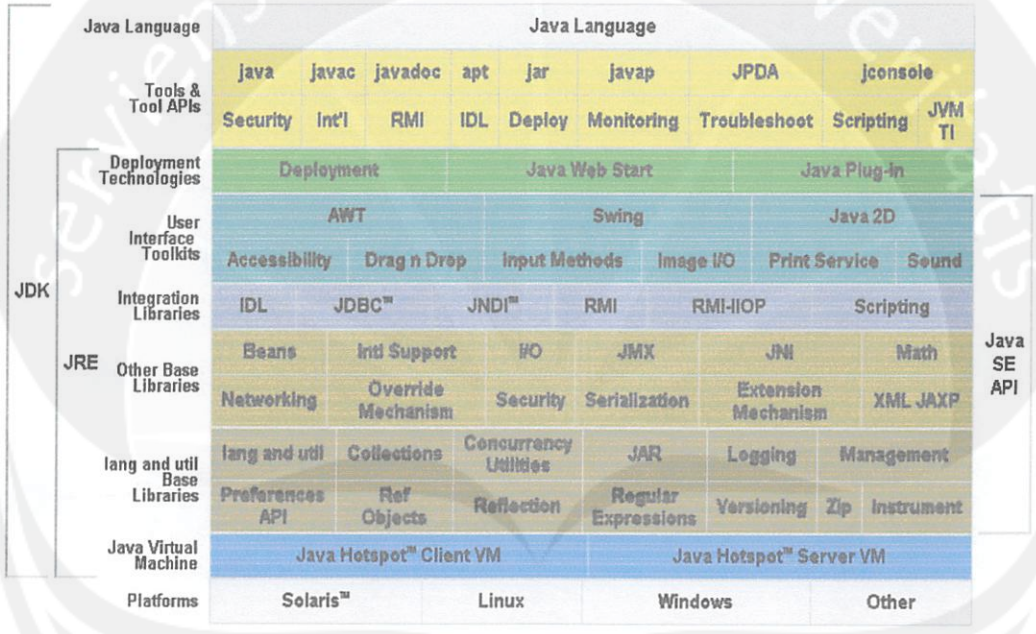
   Compiler Java (*javac*) mentranslasi kode-kode bahasa *Java* menjadi *byte-codes*.

2. *Java Virtual Machine* (JVM)

   Java Virtual Machine (JVM) adalah sebuah mesin komputer abstrak yang sering dinamakan sebagai

interpreter karena tugasnya menginterpretasikan *Java byte-codes* yang terkompilasi.

3. *Application Programming Interface* (API)
   Java API menyediakan satu set *packages* yang didistribusikan bersama dengan *Java 2 Software Development Kit* (J2SDK) sebagai *class libraries*.



Gambar 2.2 Arsitektur *Java* (java.sun.com)

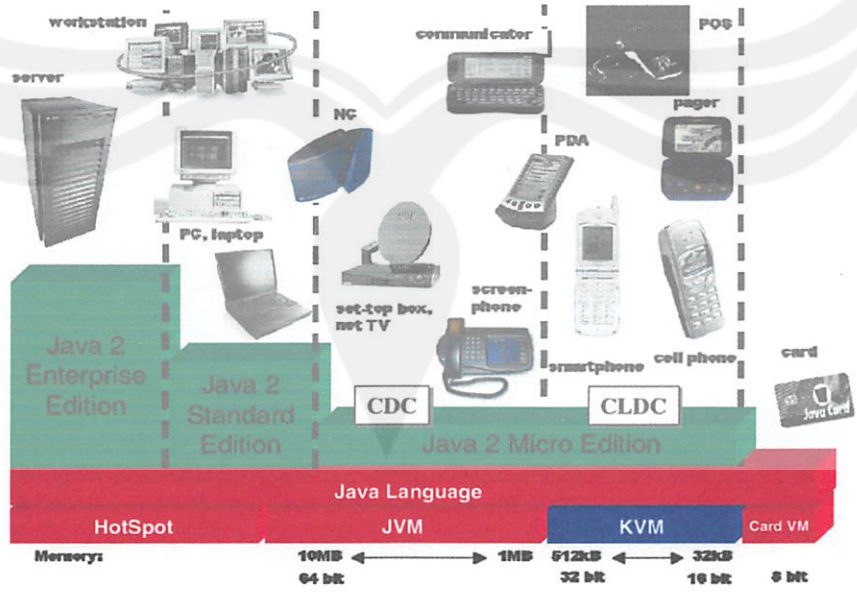
II.6.3 Edisi *Java*

Sun Microsystems telah mendefinisikan tiga buah edisi dari *Java 2* yang ditujukan untuk mesin dan sistem yang berbeda-beda, yaitu :

1. *Java 2 Standard Edition* (J2SE), yang digunakan untuk mengembangkan aplikasi-aplikasi *desktop* dan *applet* (aplikasi *Java* yang dapat dijalankan di dalam *browser web*).

2. *Java 2 Enterprise Edition* (J2EE), merupakan *superset* dari J2SE yang digunakan untuk mengembangkan aplikasi-aplikasi berskala *enterprise*, yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi *server* dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi *web* dengan menggunakan *Java Servlet* dan JSP (*Java Server Pages*), dan teknologi lainnya seperti CORBA (*Common Object Request Broker Architecture*) dan XML (*Extensible Markup Language*).
3. *Java 2 Micro Edition* (J2ME), merupakan *subset* dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh.

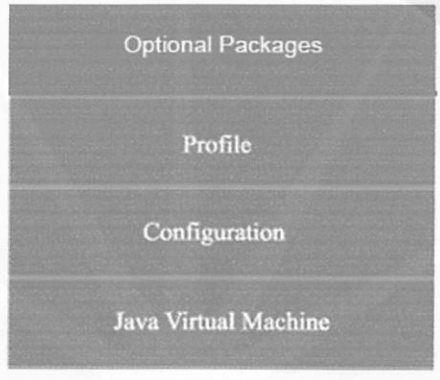
Secara umum pembagian edisi *Java* dapat digambarkan sebagai berikut :



Gambar 2.3 Pembagian Edisi *Java*

II.7 *Java 2 Micro Edition* (J2ME)

J2ME™ merupakan salah satu dari tiga *platform* teknologi *Java* yang digunakan untuk pembangunan aplikasi pada *mobile device* yang memiliki *memory* kecil seperti ponsel, *pager*, atau PDA (*Personal Digital Assistant*). Arsitektur J2ME terdiri dari *configuration*, *profile*, *virtual machine*, dan paket-paket opsional seperti yang digambarkan di bawah ini :



Gambar 2.4 Arsitektur J2ME

II.7.1 J2ME *Configuration*

Configuration mendefinisikan lingkungan kerja J2ME *runtime*. Karena setiap *handheld devices* memiliki fitur yang berbeda-beda, maka dirancanglah J2ME *configuration*, yakni menyediakan *library* standar yang mengimplementasikan fitur standar dari sebuah *handheld devices*. Ada dua kategori J2ME *Configuration* saat ini, yaitu :

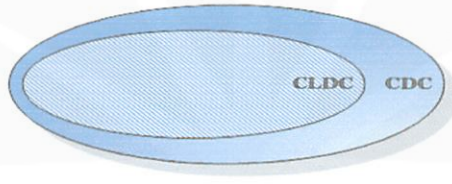
1. **CLDC (*Connected Limited Devices Configuration*)**
   Kategori ini umumnya digunakan untuk aplikasi *Java* pada *handphone* seperti Nokia, Samsung Java Phone, Motorola i85s. PDA semacam PALM, *PocketPC*, dan *two way pagers*. Umumnya perangkat tersebut hanya memiliki ukuran memory 160-512

KiloBytes. Mesin *virtual* yang digunakan untuk CLDC adalah *Kilobyte Virtual Machine* (KVM) yaitu mesin virtual yang ditujukan untuk perangkat yang memiliki ukuran *memory* kecil.

2. **CDC (*Connected Device Configuration*)**

   Kategori ini umumnya digunakan untuk aplikasi Java pada perangkat *handheld devices* dengan ukuran memory 2 *MegaBytes*. Contohnya adalah *Internet TV*, Nokia *Communicator* atau TV pada mobil.

Relasi antara CDC dan CLDC dapat digambarkan sebagai berikut :



Gambar 2.5 Relasi antara CDC dan CLDC

II.7.2 *J2ME Profile*

Profile adalah penyedia *libraries* dan API untuk para *developer* untuk mengembangkan aplikasi untuk masing-masing tipe *device*. Saat ini terdapat beberapa *profile* yang tersedia untuk kebutuhan-kebutuhan spesifik diantaranya *Mobile Information Device Profile* (MIDP), *Personal Digital Assistant Profile* (PDAP), *Foundation profile, Personal Profile, RMI Profile.*

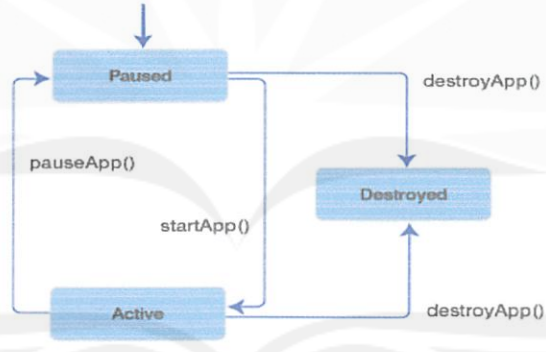
II.7.3 *Mobile Information Device Profile* (MIDP)

MIDP khusus digunakan pada *handset* dengan kemampuan CPU, *memory*, *keyboard*, dan layar terbatas, misalnya pada *handphone*, *pager*, dan PDA. MIDP

menyediakan cukup banyak *library* dan API *(Application Programming Interface)* untuk membantu pembangunan aplikasi.

II.7.3.1 *MIDlet*

MIDlet adalah aplikasi yang dibuat dengan menggunakan *Java 2 Micro Edition* dengan profile *Mobile Information Device Profile* (MIDP). *MIDlet* terdapat dalam *class javax.microedition.midlet.*. MIDlet* memiliki memiliki beberapa *state*, yaitu *Pause, Active,* dan *Destroy.* Ketika masing-masing *state* dipanggil, beberapa metoda yang bersesuaian dipanggil. State dalam *MIDlet* dapat digambarkan sebagai berikut :



Gambar 2.6 Siklus Hidup Sebuah *MIDlet*

II.7.3.2 *Graphical User Interface* (GUI)

MIDP menyediakan API untuk membangun aplikasi yang berbasis GUI yang terdapat dalam *class javax.microedition.lcdui.*. Library* ini menyediakan berbagai macam *class* dan *interface* untuk mendukung pembangunan aplikasi berbasis GUI, yaitu :

1. *Command* dan *CommandListener*

   Objek-objek *command* yang tersedia yaitu BACK, CANCEL, EXIT, HELP, ITEM, OK, SCREEN, STOP.

2. *High Level User Interface*

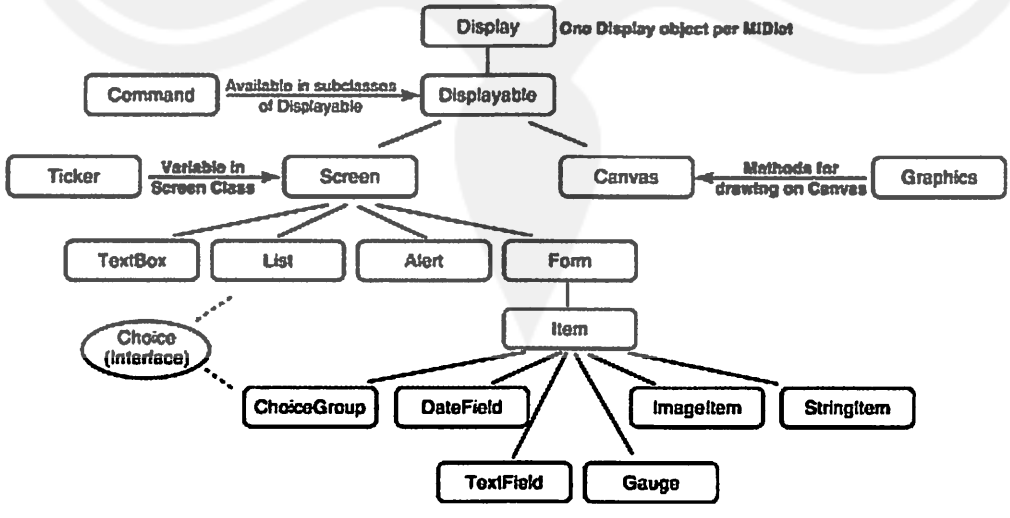
   Beberapa objek yang terdapat di dalamnya, yaitu *Screen, Form, Item, DateField, Gauge, StringItem, TextField, Choice, ChoiceGroup, Image , ImageItem, List, TextBox, Alert, AlertType,* dan *Ticker.*

3. *Low Level User Interface*

   Beberapa objek yang terdapat di dalamnya, yaitu *Canvas* dan *Graphics.*

4. *Display* dan *Displayable*

   Display merupakan objek yang dapat menerima informasi tentang tampilan layar saat ini dan juga termasuk *methods* untuk me-*request* objek yang akan ditampilkan (Muchow, 2001). Masing-masing *MIDlet* memiliki satu referensi untuk satu objek *Display* sedangkan sebuah objek display dapat menampilkan beberapa objek *Displayable.* Terdapat dua *subclass* dari *Displayable* yaitu *Screen* dan *Canvas.* Berikut ini adalah hierarki dari *class Displayable* :



Gambar 2.7 Hierarki *Class Displayable* (Muchow, 2001)

II.7.3.3 *Record Management System* (RMS)

Record Management System merupakan suatu API (*Application Programming Interface*) yang memberi aplikasi MIDP tempat penyimpanan data lokal. RMS menyediakan lingkungan menyerupai *file system* dan *Database Management System* (DBMS) yang digunakan untuk menyimpan dan memelihara data di dalam *micro device* dalam bentuk kumpulan *record-record*. RMS terdapat dalam paket *javax.microedition.rms.**. Pada saat data disimpan di dalam sebuah *record store*, data tersebut berada dalam bentuk sekumpulan larik dari tipe *byte (array byte)* yang diidentifikasi dengan suatu ID yang bersifat unik (*primary key*). Sedangkan disisi lainnya data dalam bahasa *Java* disimpan dalam bentuk objek, oleh karena itu bila akan melakukan penyimpanan ke *record store* terlebih dahulu melakukan seralisasi objek ke bentuk *array byte*. Berikut ini adalah analogi dari sebuah *record store* :

Record ID	***Data***
1	*Array of bytes*
2	*Array of bytes*
3	*Array of bytes*
...	...

Tabel 2.1 *Record Store*

RecordStore merupakan satu-satunya kelas konkrit dalam paket *javax.microedition.rms.** yang digunakan untuk membuat *record* dalam suatu aplikasi (Raharjo, 2007).

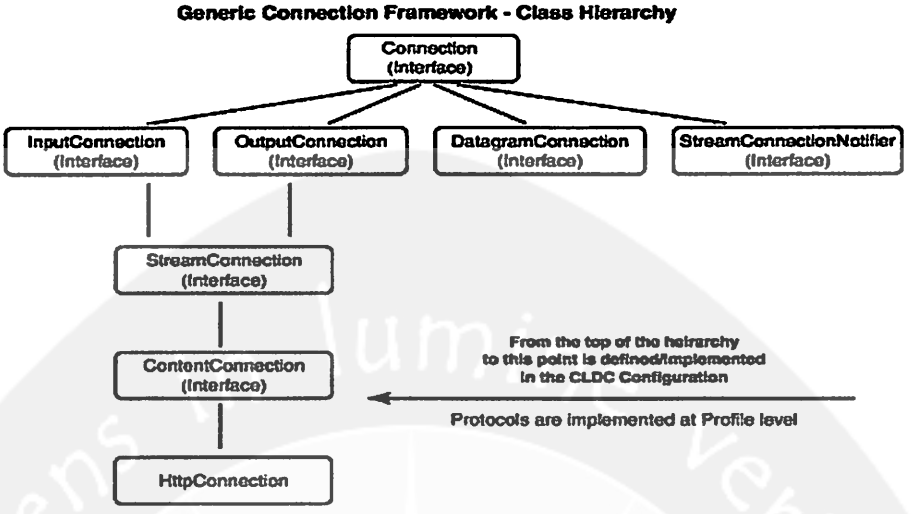
Dalam paket ini juga terdapat beberapa *interface* pendukung di dalamnya, yaitu :

1. *RecordEnumeration,* yang digunakan untuk navigasi data terhadap *record-record* dalam suatu *record store.*
2. *RecordComparator,* yang digunakan untuk membandingkan dua buah *record,* untuk mengetahui ukuran relatif dari keduanya.
3. *RecordFilter,* yang digunakan untuk melakukan filter terhadap *record-record* dalam suatu *record store* berdasarkan suatu kriteria yang didefinisikan.

Beberapa eksepsi yang juga terdapat dalam *javax.microedition.rms.** yaitu *InvalidRecordIDException, RecordStoreException, RecordStoreFullException, RecordStoreNotFoundException,* dan *RecordStoreNotOpenException.*

II.7.3.4 *Generic Connection Framework* (GCF)

GCF dikembangkan untuk menyediakan dukungan terhadap *framework i/o* dan *networking* yaitu untuk mengakses *storage* dan sistem jaringan. GCF terdapat dalam *class javax.microedition.io.**. *Class Connector* merupakan sebuah *class* yang digunakan untuk menangani penciptaan beberapa tipe koneksi, *file*, HTTP, *datagram*, dan sebagainya. Berikut ini adalah hierarki dari *class Connection* :



Gambar 2.8 Hierarki *Class Connection* (Muchow, 2001)

Untuk melakukan koneksi HTTP agar aplikasi bisa berkomunikasi dengan *web server* atau *remote device* lain yang mendukung protokol HTTP maka digunakan *class HttpConnection*. Beberapa tahapan yang digunakan untuk melakukan koneksi dan mengakses data di *web server* yaitu :

1. Membuat sebuah koneksi dengan menggunakan *class Connector*. Terdapat tiga macam mode koneksi yaitu *read only*, *write only*, dan *read and write*.
2. *Client Request*, terdapat tiga macam request method yang ditangani *HttpConnection* yaitu *GET*, *POST*, dan *HEAD*.
3. *Server Response*, terdiri dari tiga sesi yaitu *status line*, *header*, dan *body*.
4. *Connection Information*, setelah suatu koneksi terjadi terdapat beberapa *methods* yang tersedia untuk mendapatkan informasi seputar koneksi yang terbentuk misalnya *String getFile()*, *String getHost()*, dan sebagainya.

5. *Session Management*, sebuah MIDlet dapat menangani manajemen sesi dengan menggunakan *cookie*. *Method* yang dipanggil untuk mengalokasikan cookie ke dalam *persistent storage* yaitu *readCookie()*.

II.8 Penutup

Aplikasi yang akan dibangun adalah *Messaging List* yaitu merupakan sinergi antara *Instant Messaging* dan *Mailing list*. Dengan *Messaging List*, *user* tidak diharuskan sama-sama *online* secara *real time*, karena pesan-pesan yang akan dibaca di-*download* terlebih dahulu baru kemudian dapat dibaca. *Messaging List* merupakan aplikasi *two-tier* (*client-server*). *User* akan secara langsung berinteraksi dengan *Messaging List client* melalui *mobile device*, sedangkan *Messaging List server* digunakan untuk *message manager* di-*server*. *Messaging List client* dibangun dengan bahasa pemrograman J2ME dan dijalankan pada *mobile device* dengan spesifikasi CLDC 1.0 dan MIDP 2.0 sedangkan *Messaging List server* dibangun dengan bahasa pemrograman PHP dan menggunakan MySQL sebagai DBMS-nya. Aplikasi *client* dan *server* akan saling berkomunikasi menggunakan protokol HTTP dan memanfaatkan teknologi GPRS sebagai *data bearer*.