BAB II

LANDASAN TEORI

2.1 Java

Java merupakan bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya di suatu perusahaan perangkat lunak yang bernama *Sun Microsystems*, pada tahun 1991. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama "*Oak*", namun pada tahun 1995 diganti namanya menjadi "Java".

2.1.1 Pengertian Java

Menurut definisi *Sun Microsystem*, di dalam buku M. Shalahuddin dan Rosa A.S. (2010 : 1) *Java adalah nama sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer yang berdiri sendiri (standalone) ataupun pada lingkungan jaringan*.

Java berdiri di atas sebuah mesin penterjemah (*interpreter*) yang diberi nama *Java Virtual Machine* (JVM). JVM inilah yang akan membaca kode bit (*bytecode*) dalam file *.class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu bahasa Java disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada system operasi tersebut terdapat JVM. Alasan utama pembentukan bahasa Java adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan di berbagai macam perangkat elektronik, sehingga Java harus bersifat tidak bergantung pada platform (*platform independent*). Itulah yang menyebabkan dalam dunia pemrograman Java dikenal adanya istilah '*write once, run everywhere*', yang berarti kode program hanya ditulis sekali, namun dapat

dijalankan di bawah kumpulan pustaka (*platform*) manapun, tanpa harus melakukan perubahan kode program.

2.1.2. Arsitektur Java

Secara arsitektur, Java tidak berubah sedikitpun sejak awal mula bahasa tersebut dirilis. *Compiler* Java (yang disebut dengan javac atau *Java Compiler*) akan mentransformasikan kode-kode dalam bahasa Java ke dalam suatu kode bit. Dimana *bytecode* adalah sekumpulan perintah hasil kompilasi yang kemudian dapat dieksekusi melalui sebuah mesin komputer abstrak, yang disebut dengan JVM (*Java Virtual Machine*). JVM juga sering dinamakan sebagai *interpreter*, karena sifatnya yang selalu menerjemahkan kode-kode yang tersimpan dalam kode bit dengan cara baris demi baris. Untuk menjalankan program Java, maka file dengan ekstensi *.java* harus dikompilasi menjadi file kode bit. Dimana untuk menjalankan kode bit tersebut dibutuhkan JRE (*Java Runtime Environment*) yang memungkinkan pemakai untuk menjalankan program Java, hanya menjalankan, tidak untuk membuat kode baru lagi. JRE berisi JVM dan pustaka Java yang digunakan.

2.2 Java 2

Sun Microsystems telah mendefinisikan tiga buah edisi dari Java 2, yaitu sebagai berikut :

1. ***Java 2 Standard Edition* (J2SE)**, adalah inti dari bahasa pemrograman Java. JDK merupakan salah satu perangkat (*tool*) dari J2SE untuk mengkompilasi dan menjalankan program Java. Di dalamnya terdapat

perangkat untuk mengkompilasi program Java dan JRE. J2SE ini digunakan pada perangkat keras seperti layar komputer (*desktop*).

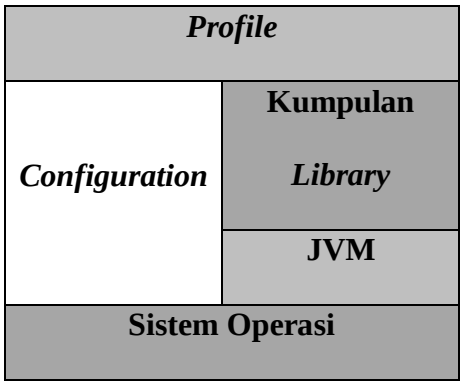
2. ***Java 2 Enterprise Edition*** **(J2EE)**, merupakan kumpulan tertinggi (*superset*) dari J2SE yang memperbolehkan kita untuk mengembangkan aplikasi-aplikasi berskala besar (*enterprise*) karena dijalankan pada jaringan komputer.
3. ***Java 2 Micro Edition*** **(J2ME)**, merupakan kumpulan bagian (*subset*) dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh. Paket J2ME digunakan pada perangkat yang memiliki kapasitas memori kecil seperti telepon selular, pager atau PDA.

2.3 *Java 2 Micro Edition* (J2ME)

2.3.1 Pengenalan *Java 2 Micro Edition* (J2ME)

Java2 *Micro Edition* atau yang biasa disebut J2ME adalah lingkungan pengembangan yang didesain untuk meletakan perangkat lunak Java pada barang elektronik beserta perangkat pendukungnya. Pada J2ME, jika perangkat lunak berfungsi baik pada sebuah perangkat maka belum tentu juga berfungsi baik pada perangkat lainnya. J2ME membawa Java ke dunia informasi, komunikasi, dan perangkat komputasi selain perangkat komputer *dekstop* yang biasanya lebih kecil dibandingkan perangkat komputer *dekstop.* J2ME biasa digunakan pada telepon seluler, *pager, personal digital assistants* (PDA's) dan sejenisnya.

J2ME adalah bagian dari J2SE, karena itu tidak semua *library* yang ada pada J2SE dapat digunakan pada J2ME. Tetapi J2ME mempunyai beberapa *library* khusus yang tidak dimiliki J2SE. Arsitektur J2ME dapat dilihat pada gambar 2.1

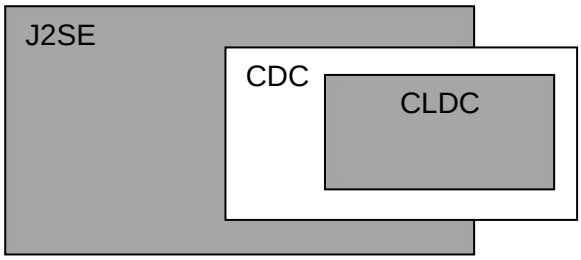


Gambar 2.1 Arsitektur J2ME

Teknologi J2ME juga memiliki beberapa keterbatasan, terutama jika diaplikasikan pada ponsel. J2ME sangat tergantung pada perangkat (*device*) yang digunakan, bisa dari segi merk ponsel, maupun kemampuan ponsel, dan dukungannya terhadap teknologi J2ME. Misalnya, jika sebuah ponsel tidak memiliki kamera maka jelas J2ME pada ponsel tersebut tidak dapat mengakses kamera. Keterbatasan lainnya adalah pada ukuran aplikasi, karena memori pada ponsel sangat terbatas. Sebagian besar ponsel tidak mengijinkan aplikasi J2ME menulis pada *file* karena alasan keamanan.

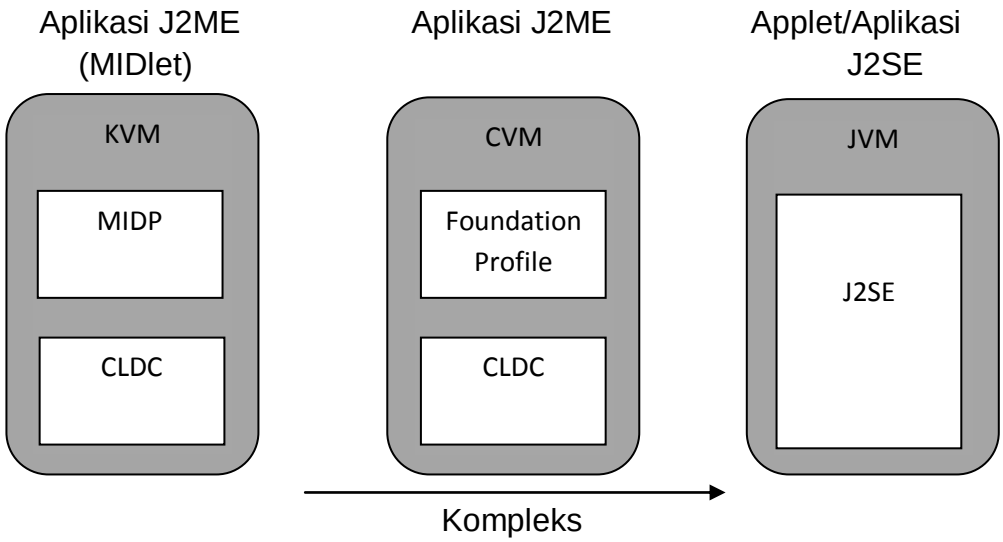
Configuration merupakan Java *library* minimum dan kapabilitas yang dipunya oleh para pengembang J2ME, yang maksudnya sebuah *mobile device* dengan kemampuan java akan dioptimalkan untuk menjadi sesuai. *Configuration* hanyalah mengatur hal-hal tentang kesamaan sehingga dapat dijadikan ukuran ukuran kesesuaian antar-*device.* Misalnya sebuah lampu sepeda dirancang sedemikian rupa sehingga dapat digunakan oleh berjenis-jenis sepeda. Dalam

J2ME telah didefinisikan dua buah konfigurasi yaitu CLDC (*Connected Limited Device Configuration*) untuk perangkat kecil dan CDC (*Connected Device Configuration*) untuk perangkat yang lebih besar. Lingkup CLDC dan CDC dapat dilihat pada gambar 2.2



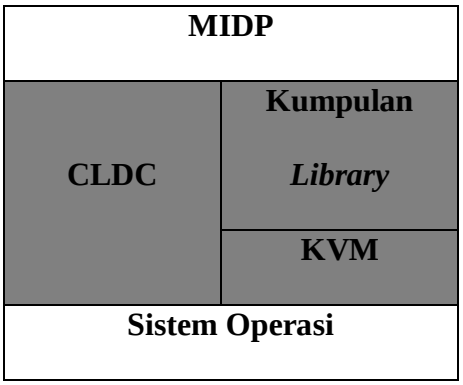
Gambar 2.2 Lingkup *Configuration*

Profile berbeda dengan *Configuration, Profile* membahas sesuatu yang spesifik untuk sebuah perangkat. Sebagai contoh misalnya, sebuah sepeda dengan merk tertentu tentu mempunyai spesifik dengan sepeda lain. Dalam J2ME terdapat dua buah *Profile* yaitu MIDP dan *Foundation Profile.* Keterhubungan antara *Configuration* dan *Profile* yang ada pada J2ME beserta jenis mesin virtualnya dapat dilihat pada gambar 2.3



Gambar 2.3 Hubungan J2ME dan J2SE

2.3.2 *Connected Limited Device Configuration* (CLDC)



Gambar 2.4 Arsitektur J2ME Pada (CLDC)

CLDC atau *Connected Limited Device Configuration* adalah perangkat dasar dari J2ME, spesifikasi dasar yang berupa *library* dan API yang diimplementasikan pada J2ME, seperti yang digunakan pada telepon selular, *pager*, dan PDA. Perangkat tersebut dibatasi dengan keterbatasan memori, sumber daya, dan kemampuan memproses, Spesifikasi CLDC pada J2ME adalah spesifikasi minimal dari *pacage*, kelas, dan sebagian fungsi *Java Virtual Machine* yang dikurangi agar dapat diimplementasikan dengan keterbatasan sumber daya pada alat – alat tersebut, JVM yang digunakan disebut KVM (*Kilobyte Virtual Machine*).

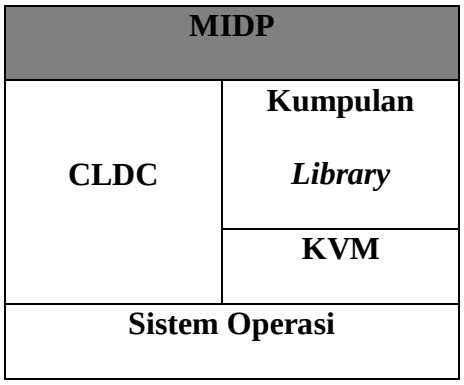
2.3.3 *Connected Device Configuration* (CDC)

CDC atau *Connected Device Configuration* adalah spesifikasi dari konfigurasi J2ME. CDC merupakan komunitas proses pada java yang memiliki standardisasi. CDC terdiri dari *virtual machine* dari kumpulan *library* dasar untuk dipergunakan pada *profile* industri. Implementasi CDC pada J2ME adalah *source code* yang menyediakan sambungan dengan macam-macam *platform*.

Tabel 2.1 perbandingan CLDC dengan CDC

CLDC	CDC
Mengimplementasikan sebagian dari J2SE	Mengimplementasikan seluruh fitur J2SSE
JVM yang digunakan adalah KVM	JVM yang digunakan adalah CVM
Digunakan pada perangkat genggam (handphone, PDA, twoway pager) dengan memori terbatas (160-512 KB)	Digunakan pada perangkat genggam (Internet TV, Nokia Comunicator, car TV) dengan memori minimal 2MB
Prosesor : 16/32 bit	Prosesor : 32 bit

2.3.4 *Mobile Information Device Profile* (MIDP)



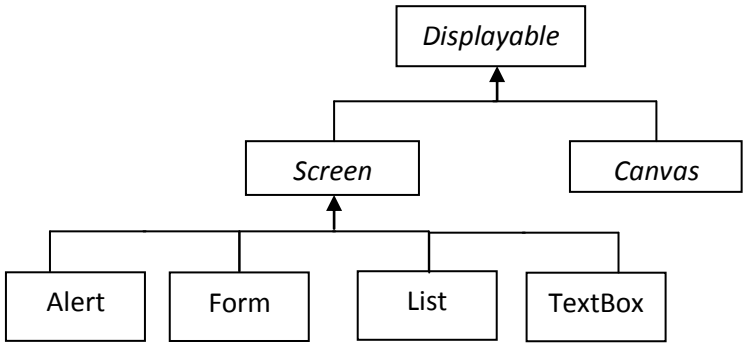
Gambar 2.5 Arsitektur J2ME Pada (MIDP)

MIDP atau *Mobile Information Device Profile* adalah spesifikasi untuk sebuah profil J2ME. MIDP memiliki lapisan di atas CLDC, API tambahan untuk daur hidup aplikasi, antarmuka, jaringan, dan penyimpanan persisten. Pada saat ini terdapat MIDP 1.0 dan MIDP 2.0. Fitur tambahan MIDP 2.0 dibanding MIDP 1.0 adalah API untuk multimedia. Pada MIDP 2.0 terdapat dukungan memainkan *tone, tone sequence,* dan file WAP walaupun tanpa adanya *mobile media API* (MMAPI).

Tabel 2.2 perbandingan MIDP 1.0 dengan MIDP 2.0

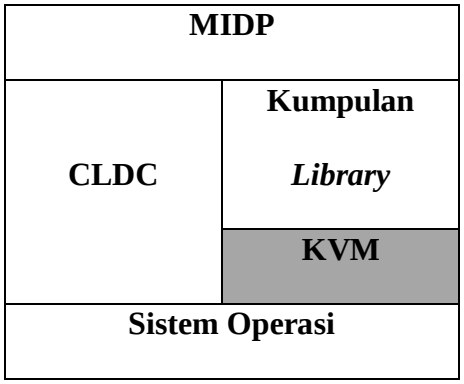
Spesifikasi	MIDP 1.0	MIDP 2.0
Display	96 x 54	96 x 54
Kedalaman *Display*	1-bit	1-bit
Bentuk piksel (rasio aspek)	Mendekati 1:1	Mendekati 1:1
Input	Keyboard dan *touch screen*	Keyboard dan *touch screen*
Memori	128 KB memori *non-volatile* untuk komponen MIDP 8 KB memori *non-volatile* untuk data *parsistence* yang dibuat oleh aplikasi 32 KB memori *volatile* untuk JRE	256 KB memori *non-volatile* untuk komponen MIDP 8KB memori *non-volatile* untuk data *parsistence* yang dibuat oleh aplikasi 128 KB memori *volatile* untuk JRE
Jaringan	Dua arah, tanpa kabel (*wireles*)	Dua arah, tanpa kabel (*wireles*)
Library J2ME yang bukan merupakan *library* J2SE	javax.microedition.lcdui, javax.microedition.midlet, javax.microedition.rms	javax.microedition.lcdui, javax.microedition.midlet, javax.microedition.rms, javax.microedition,lcdui.game, javax.microedition.media, javax.microedition.pki
Multimedia		Memiliki kemampuan untuk memainkan file multimedia (suara dan video)

MIDP *User Interface* API memiliki API level rendah. API level rendah berbasiskan penggunaan dari kelas abstrak Canvas, sedangkan kelas API level tinggi antara lain Alert, Form, List, dan TexBox yang merupakan extensi dari kelas abstrak Screen. API level rendah lebih memberikan kemudahan kepada pengembang untuk memodifikasi sesuai dengan kehendaknya, sedangkan API level tinggi biasanya hanya memberikan pengaksesan yang terbatas. Arsitektur antarmuka MIDP dapat dilihat pada gambar 2.6



Gambar 2.6 MIDP *User Interface*

2.3.5 *Kilobyte Virtual Machine* (KVM)



Gambar 2.7 Arsitektur J2ME Pada (KVM)

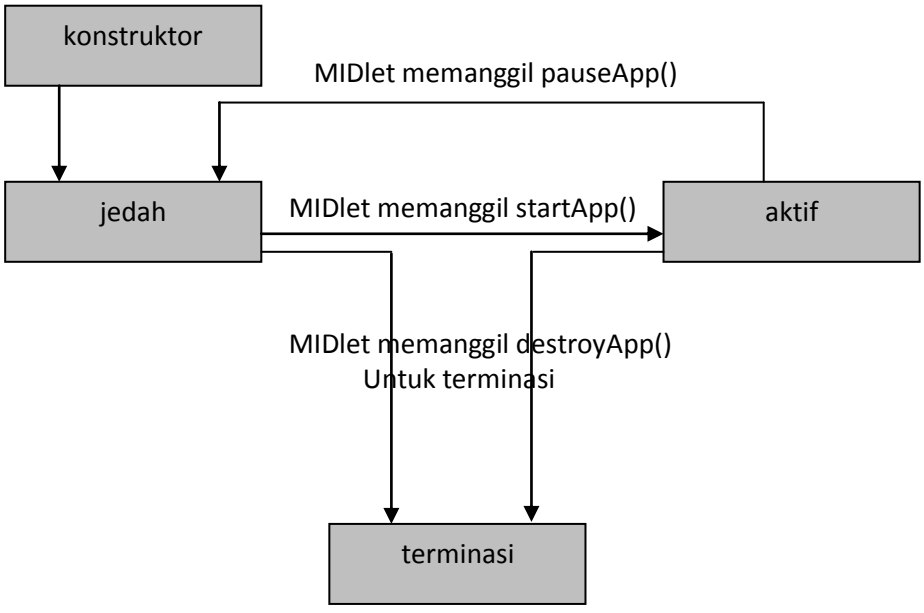
KVM adalah paket JVM yang didesain untuk perangkat yang kecil. KVM mendukung sebagian dari fitur-fitur JVM, seperti misalnya KVM tidak mendukung operasi *floating-point* dari finaliasi objek. KVM diimplementasikan dengan menggunakan C sehingga sangat mudah beradaptasi pada tipe *flatform* yang berbeda.

2.3.6 MIDlet

MIDlet adalah aplikasi yang ditulis untuk MIDP. Aplikasi MIDlet adalah bagian dari kelas javax.microedition.midlet.MIDlet yang didefinisikan pada MIDP. MIDlet berupa sebuah kelas abstrak yang merupakan subkelas dari bentuk dasar aplikasi sehingga antarmuka antara aplikasi J2ME dan aplikasi manajemen pada perangkat dapat terbentuk.

2.3.6.1 Daur Hidup MIDlet

MIDlet terdiri dari beberapa metode yang harus ada, yaitu constructor ( ), protected void startApp ( ) throws MIDletStateChangeException, protected void pauseApp ( ), protected void destroyApp(boolean unconditional) throws MIDletStateChangeException. Alur hidup MIDlet dapat dilihat pada gambar 2.8



Gambar 2.8 Alur Hidup MIDlet

Ketika MIDlet dijalankan maka akan diinisialisasi dengan kondisi *pause* dan dijalankan pauseApp() , Kondisi berikutnya adalah fungsi MIDlet dijalankan, yaitu startApp(). *Metode* yang ada tersebut diimplementasikan sebagai protected, hal ini dimaksudkan agar MIDlet lain tidak dapat memanggil *metode* tersebut. Pada saat pemakai keluar dari MIDlet, maka *metode* destroyApp() akan dijalankan sebelum MIDlet benar-benar tidak berjalan lagi, destroyApp() akan memanggil notifyDestroyed() , dan notifyDestroyed() akan memberitahu *platform* untuk menterminasi MIDlet dan membersihkan semua sumber daya yang mengacu pada MIDlet.

Dalam implementasinya, MIDlet memiliki struktur direktori sebagai berikut:

1. src

   menyimpan *source code* untuk MIDlet dan kelas lain yang diperlukan

2. res

   menyimpan sumber daya yang dibutuhkan oleh MIDlet, seperti misalnya gambar *icon*

3. lib

   menyimpan *file* JAR atau ZIP yang berisi *library* tambahan yang dibutuhkan MIDlet

4. bin

   menyimpan *file* JAR, JAD, dan *file* manifest yang berisi muata komponen MIDlet

2.3.6.2 Atribut MIDlet

MIDlet juga memiliki Atribut *package* sebagai berikut :

Tabel 2.3 Atribut *Package*

Nama Atribut	Nilai dan Fungsi
MIDlet – Name	Nama MIDlet yang juga sebagai nama untuk *file* JAR-nya. Misalnya WirelessTrader
MIDlet – Version	Nomor versi dari MIDlet yang juga sebagai nomor versi pada *file* JAR-nya. Misalnya 1.0.1
MIDlet – Vendor	Nama *provider* MIDlet. Misalnya Wireles JavaInc.
MIDlet – *n*	Atribut yang mendeskripsikan MIDlet. Nilai *n* diganti dengan nilai numerik dimulai dari angka 1. Format nilai dari atribut ini terkain dengan atribut yang dideskripsikan selanjutnya
MicroEdition – Profile	Versi dari spedifikasi MIDlet yang dapat berjalan. Dapat lebih dari satu versi yang dipisah dengan spasi, contoh dari atribut ini adalah MIDP – 1.0
MicroEdition - Configuration	Konfigurasi J2ME yang dibutuhkan untuk MIDlet.
MIDlet – Description	Deskripsi MIDlet. Misalnya MIDlet untuk koneksi Internet
MIDlet – Icon	*Icon* yang digunakan MIDlet. Misalnya wireles.png
MIDlet – Info – URL	URL dari *file* yang berisi informasi mengenai MIDlet. Misalnya http://www.wireless.com/info.html
MIDlet – Data – Size	Nilai minimum dari besarnya tempat penyimpanan persisten yang dibutuhkan (dalam *byte*) tidak termasuk yang dibutuhkan untuk menginstal aplikasi. Misalnya 512.
MIDlet – Jar – URL	URL dari *file* JAR. Misalnya http://www.wireless.com/MIDlet.jar
MIDlet – Jar – Size	Ukuran *file* JAR dalam *byte*. Misalnya 10312.
MIDlet – Install – Notify	Sebuah URL yang digunakan untuk melaporkan sukses atau gagalnya penginstalan MIDlet dari *Remote Server*.
MIDlet – Delete – Confirm	Pesan yang ditampilkan ke pamakai sebelum MIDlet dihapus dari alat di mana MIDlet diinstal.
MIDlet –Specificattributes	Pengembang MIDlet dapat menyediakan konfigurasi minimum untuk MIDlet dengan memasukan atribut ini.

2.3.7 JAD (*Java Application Descriptor*)

Digunakan untuk mendeskripsikan isi aplikasi untuk keperluan pemetaan. *File* JAD berisi deskripsi *file* JAR (*Java Archive*) dan pemetaan atribut MIDlet, sedangkan *file* JAR berisi kumpulan kelas dan resource.

2.4 J2ME *Wireles Toolkit*

J2ME *Wireles Toolkit* adalah kakas yang menyediakan lingkungan *emulator*, dokumentasi beserta contoh-contoh aplikasi Java untuk perangkat kecil (*small device*). J2ME WTK berbasiskan pada CLDC dan MIDP. J2ME WTK adalah program yang meniru kerja ponsel yang mendukung MIDP atau yang disebut *emulator*. Oleh karena itu, belum tentu MIDlet yang berjalan di *emulator* juga berjalan pada ponsel yang sebenarnya, karena juga bergantung pada kemampuan dan kapasitas ponsel yang digunakan.

2.5 Komponen-komponen J2ME

2.5.1 *Display*

Display merupakan objek yang merepersentasikan pengelola layar pada perangkat. Pada sebuah MIDlet hanya terdapat satu objek *display*. Objek *display* menyediakan metode untuk menggambar dan menampilkan elemen antarmuka grafis pada layar. Objek *display* juga menyediakan metode untuk mengetahui properti layar perangkat seperti apakah layar perangkat mendukung layar berwarna atau tidak.

Pada *library* J2ME, objek *Display* berada pada kelas javax.microedition.lcdui.Display yang memiliki metode-metode sebagai berikut:

Tabel 2.4 Metode Objek *Display*

Metode	Keterangan
Static Diplay **getDisplay**(MIDlet m)	Mengambil objek *display* untuk MIDlet yang bersangkutan.
Displayable **getCurrent**()	Merupakan metode untuk mengakses objek *Displayable* yang sedang

	ditampilkan.
void **setCurrent**(Alert alert, Displayeble next Displayable)	Menampilkan sebuah *Alert* dengan diikuti objek-objek *Displayable* yang menjadi parameter masukan metode ini.
void **setCurrent**(Displayeble next Displayable)	Merupakan metode untuk menampilkan *Displayable* yang menjadi parameter masukan metode ini.
boolean **isColor**()	Mengembalikan *true* jika perangkat mendukung warna.
int **numColors**()	Mengembalikan jumlah warna yang didukung oleh perangkat.
void **vibrate** (int milidetik)	Menggetarkan telepon genggam
void **flashBacklight** (int milidetik)	Menyalakan lampu telepon genggam

2.5.2 *Displayable*

Displayable adalah kelas abstrak dari antarmuka pemakai (*user interface)*. *Displayable* memiliki dua buah subkelas yaitu Canvas sebagai antarmuka level rendah dan *screen* sebagai antarmuka level tinggi. Pada sekali waktu, hanya dapat ditampilkan satu buah objek *Displayable* yang disebut sebagai *current Displayable*. *Displayable* berbeda dengan *Display*, jika dalam sebuah MIDlet hanya boleh ada sebuah *Display*, lain dengan *Displayable*, pada sebuah MIDlet boleh terdapat lebih dari satu *Displayable*.

2.5.3 *Screen*

creen adalah kelas diatas semua antarmuka pemakai level tinggi atau bisa disebut *Superclass*. Sebagian besar komponen *Screen* tidak mengijinkan penambahan komponen lain dan penghapusan pada komponen tersebut, komponen-komponen itu antara lain *List*, *TextBox*, dan *Alert*, misalnya jika

sebuah *TextBox* yang ditampilkan maka pengembang tidak dapat lagi menambahkan komponen lain dengan level sejajar *TextBox.*

Objek *Screen* didefinisikan pada *library* J2ME javax.microedition.lcdui.Screen yang memiliki metode-metode sebagai berikut

Tabel 2.5 Metode Objek *Screen*

Metode	Keterangan
String getTitle()	Mengembalikan nilai *title* yang terasosiasi dengan *Screen.*
void setTitle(String s)	Mengeset *title Screen.*
Ticker getTicker ()	Mengembalikan *pointer* Ticker yang terasosiasi dengan *Screen.*
void setTicker(Ticker ticker)	Mengeset *Ticker* pada *Screen.*

2.5.4 *Font*

Sebuah *font* digunakan pada antarmuka pemakai level rendah. *Font* digunakan untuk mendefinisikan mode sebuah karakter atau *string* ke layar. *Font* ini hanya dapat digunakan pada antarmuka pemakai level rendah.

Tabel 2.6 Atribut-atribut *Font* pada J2ME

Mode (*Style)*	Jenis(*Face)*	Ukuran (*Size)*
STYLE_BOLD STYLE_ITALIC STYLE_UNDERLINED STYLE_PLAIN	FACE_MONOSPACE FACE_PROPORTIONAL FACE_SYSTEM	SIZE_LARGE SIZE_SMALL SIZE_MEDIUM

2.5.5 *Command*

Command adalah objek yang memungkinkan pemakai melakukan aksi. Objek Fungsi *Command* sama dengan tombol (*button*) pada aplikasi *dekstop* pada komputer. *Command* membutuhkan antarmuka (*Interface*) *CommandListener* untuk menangkap *even* dari *Command*. Saat membuat sebuah aplikasi J2ME jangan lupa membuat sebuah *Command* untuk keluar dari aplikasi karena jika ponsel tidak mendukung keluar aplikasi dengan sendirinya maka ponsel harus dimatikan untuk keluar dari aplikasi J2ME.

Tabel 2.7 Nilai-nilai Parameter *Command*

Tipe Nilai Parameter *Command*	**Keterangan**
OK	Mengimlikasikan persetujuan pemakai untuk aksi yang akan dikerjakan, biasanya *command* ini akan diletakan pada tempat yang mudah dilihat oleh pemakai.
BACK	Mengganti layar yang ditampilkan menjadi layar sebelumnya yang ditampilkan.
CANCEL	Membatalkan aksi yang akan dilakukan
STOP	Menghentikan proses yang sedang berjalan
EXIT	Menerminasi MIDlet atau keluar dari aplikasi
HELP	Meminta bantuan untuk memperjelas pemakaian kepada pemakai
SCREEN	Terkait dengan fungsi dari halaman layar yang sedang ditampilkan
ITEM	Mengindikasikan *Command* terasosiasi dengan komponen lain

2.5.6 *List*

List menyediakan fungsi memilih elemen dalam *List* kepada pemakai. Elemen tersebut dapat berupa teks, *string*, ataupun gambar.

Komponen *List* mengimplementasikan antarmuka (*Interface*) Choice yang mendefinisikan tiga tipe yang dapat diimplementasikan pada *List.* Ketiga tipe tersebut terdapat pada javax.microedition.lcdui.Choice. ketiga tipe itu adalah :

Tabel 2.8 Tipe *List*

Tipe Choice	Keterangan
Choice. EXCLUSIVE	Hanya dapat memilih satu pilihan.
Choice. MULTIPLE	Dapat tidak memilih atau memilih lebih dari satu pilihan.
Choice. IMPLICIT	Dengan memilih sebuah elemen maka sebuah even akan digenerasi

2.5.7 *Alert*

Alert adalah sejenis pesan yang tampil di layar yang menampilkan teks maupun gambar ke layar yang berguna untuk menginformasikan sesuatu kepemakai.

Tabel 2.9 Tipe *Alert*

Tipe Alert	Kegunaan
ALARM	Pemberitahuan kepada pemakai jika telah memenuhi kondisi yang sudah diatur sebelumnya.
CONFIRMATION	Konfirmasi sebuah aksi kepada pemakai.
ERROR	Mengindikasikan bahwa telah terjadi error
INFO	Informasi kepada pemakai
WARNING	Peringatan kepada pemakai

2.5.8 *Ticker*

Objek *Ticker* dapat berasosiasi dengan objek subkelas dari *screen. Ticker* merupakan objek yang berupa tulisan berjalan. Arah dan kecepatan dari *Ticker* tidak dapat diatur secara manual, karena telah diatur oleh sistem dan *Ticker* yang sedang berjalan tidak dapat dihentikan oleh aplikasi.

Metode-metode yang ada pada javax.microedition.lcdui.Ticker sebagai *library* yang mendefinisikan *Ticker* sebagai berikut :

Tabel 2.10 Metode-metode *Ticker*

Tipe Choice	Keterangan
Ticker (String str)	Membuat *Ticker* baru dengan teks str.
String getString()	Mengembalikan nilai teks yang terasosiasi dengan *Ticker*
Void setString (String str)	Mengeset teks yang terasosiasi dengan *Ticker* menjadi str.

2.5.9 *Form*

Form dapat dianggap sebagai halaman untuk memasukan data. *Form* dapat terdiri dari komponen-komponen yang biasa disebut dengan *item*. Kumpulan *item* yang ada pada sebuah *form* pada konsepnya disimpan di dalam *array*, sehingga pengaksesannya dapat dilakukan dengan menggunaka indeks.

2.5.10 *ChoiceGroup*

ChoiceGroup adalah kelompok yang dapat dipilih dengan menyeleksi pilihan yang ada dalam kelompok tersebut. *ChoiceGroup* memiliki kesamaan dengan *List* yaitu sebagai objek yang menyediakan fungsi memilih elemen bagi pemakai. Namun kedua objek ini berbeda, *List* merupakan subkelas dari *screen* sedangkan *ChoiceGroup* adalah subkelas dari *Item*, *ChoiceGroup* tidak dapat menggunakan tipe *Choice* IMPLISIT sedangkan *List* bisa, dan perubahan *even* yang terjadi pada *ChoiceGroup* ditangkap oleh ItemStateListener dengan memanggil metode ItemStateChanged() sedangkan *List* perubahan *even* ditangkap oleh *Command* dengan memanggil metode commandAction ().

Tabel 2.11 Tipe-tipe *ChoiceGroup*

Tipe Choice	Keterangan
Choice. EXCLUSIVE	Hanya dapat memilih satu pilihan.
Choice. MULTIPLE	Dapat tidak memilih atau memilih lebih dari satu pilihan.
Choice. POPUP	Tampilan menu *ChoiceGroup* akan muncul jika diklik, seperti halnya popup menu.
Choice. TEXT_WRAP_OFF	Teks menu menggunakan fungsi *warp* dan dapat memilih lebih dari satu pilihan.
Choice. TEXT_WRAP_ON	Teks menggunakan fungsi *warp* dan hanya dapat memilih satu pilihan.

2.5.11 *Image*

Sebuah *image* menyimpan sebuah data grafis gambar. *Image* dapat ditambahkan pada antarmuka pemakai level tinggi seperti *Form* dengan menggunakan metode add() , sedangkan pada antarmuka level rendah seperti

Canvas dengan menggunakan metode drawImage (). Pada antarmuka pemakai level tinggi, hanya dapat menggunakan gambar yang bersifat *immutable* yaitu gambar yang tidak dapat dimodifikasi setelah dibuat. Tidak dapat dimodifikasikan dalam artian *Item* didalamnya tidak dapat diubah secara langsung menggunakan *Index* atau *reference* ke *Item* tersebut, tetapi harus dengan rekontruksi ulang. Sedangkan pada antarmuka level rendah dapat menggunakan *mutable* yang biasanya disimpan pada memori.

Metode-metode yang ada pada javax.microedition.lcdui.Image, *library* yang mendefinisikan *Image*, sebagai berikut:

Tabel 2.12 *Library Image*

Metode	**Keterangan**
Static Image **createImage** (String name)	Membuat sebuah gambar *immutable* dari direktori resource.
Static Image **createImage** (Image source)	Membuat sebuah gambar *immutable* dari sebuah *Image* lain
Static Image **createImage** (byte [] imageData, int imageOffset, int imageLength)	Membuat sebuah gambar *immutable* dari *array* yang menyimpan datagambar mulai dari indeks imageOffset dengan panjang imageLength.
Static Image **createImage** (int width, int height)	Membuat gambar *immutable* dengan lebar width dan tinggi heigth.
Graphics **getGraphics** ()	Mengembalikan *Graphics* yang digunakan untuk menggambar gambar *mutable.*
Int getHeight ()	Mengembalikan nilai tinggi gambar
Int getWidth ()	Mengembalikan nilai lebar gambar
Boolean isMutable ()	Mengembalikan *true* jika gambar *mutable.*

2.5.12 *ImageItem*

ImageItem merupakan objek untuk menampilkan gambar seperti objek *Image*, hanya saja *ImageItem* dilengkapi dengan adanya fasilitas pengaturan *layout* atau struktur tempat gambar pada layar.

ImageItem memiliki beberapa *layout* atau cara menyusun gambar pada layar, diantaranya sebagai berikut :

Tabel 2.13 *Layout ImageItem*

Layout	Keterangan
ImageItem.LAYOUT_DEFAULT	Gambar akan disusun berdasarkan susunan standar dari platform.
ImageItem.LAYOUT_LEFT	Gambar akan disusun rata kiri.
ImageItem.LAYOUT_RIGHT	Gambar akan disusun rata kanan.
ImageItem.LAYOUT_CENTER	Gambar akan disusun pada bagian tengah layar.
ImageItem.LAYOUT_NEWLINE_BEFORE	Gambar akan digambar pada layar setelah adanya baris baru.

2.5.13 *StringItem*

Sebuah *StringItem* menampilkan sebuah label statis dan sebuah pesan yang berupa teks.

Metode-metode yang ada pada javax.microedition.lcdui.StringItem, sebagai *library* yang mendefinisikan *StringItem*, sebagai berikut:

Tabel 2.14 Metode-metode *StringItem*

Metode	Keterangan
StringItem (String label, String text)	Membuat *StringItem* baru.
String **getText**()	Mengembalikan nilai teks pada *StringItem.*
void **setText** (String text)	Mengeset teks pada *StringItem.*

2.6 UML (*Unified Modelling Language*)

Menurut Prabowo Pudjo Widodo dan Herlawati. (2011:6) UML singkatan dari *Unified Modelling Language* yang berarti sebuah bahasa untuk pemodelan standar. (Chonoles, 2003:bab 1) mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantik. Berikut diagram-diagram UML :

1. *Use case diagram*
2. *Class diagram*
3. *Activity diagram*
4. *Sequence diagram*
5. *Component diagram*
6. *Deployment diagram*

2.6.1 *Use Case Diagram*

Use case diagram menggambarkan fungsi tertentu dalam suatu sistem berupa komponen, kejadian atau kelas. Dan *Use Case* juga dapat diartikan sebagai urutan langkah-langkah yang secara tindakan saling terkait (sekenario), baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal.

2.6.2 *Class Diagram*

Class Diagram atau Diagram Class adalah inti dari proses pemodelan objek. Baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini. *Forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model.

2.6.3 *Activity Diagram*

Activity Diagram atau aktivitas diagram lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit. Diagram ini tidak hanya memodelkan *software* melainkan memodelkan model bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam bentuk kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram akivitas merepresantasikan pemanggilan suatu fungsi tertentumisalnya *call*.

2.6.4 *Sequence Diagram*

Sequence diagram menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus berasosiasi dengan *use case*. *Sequence diagram* memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*.

2.6.5 *Component Diagram*

Manfaat *component diagram* atau diagram komponen adalah bila ada salah satu komponen yang rusak atau tidak sesuai dengan tujuan sistem, kita tinggal mengganti komponen itu dengan komponen yang lain.

2.6.6 *Deployment Diagram*

Model diagram *deployment* bagian-bagian perangkat lunak suatu sistem ke perangkat keras yang akan mengeksekusinya. Elemen-elemen perangkat lunak seperti komponen, kelas, paket dan sebagainya dimanifestasikan mengguanakan artifak serta dipetakan keperangkat keras yang akan menjalankannya dengan titik (*nodes*).

2.7 Rambu-rambu Lalulintas

Aryuanto at al (1979 : 1) "Rambu-rambu lalu lintas umumnya mempunyai warna yang kontras dengan lingkungan sekitarnya, sehingga mudah dilihat dan menarik perhatian pengendara. Rambu-rambu lalu lintas dibedakan menjadi beberapa jenis, seperti : a) rambu pengatur (rambu batas kecepatan, rambu dilarang masuk, dll); b) rambu peringatan (rambu peringatan ada pekerjaan jalan, rambu peringatan perlintasan jalan, dll); c) rambu petunjuk (rambu petunjuk jalan dan informasi rute jalan, dll".

a. Rambu peringatan

   Rambu yang memperingatkan adanya bahaya agar para pengemudi berhati-hati dalam menjalankan kendaraannya. Misalnya : menunjukkan adanya lintasan kereta api atau adanya simpangan berbahaya bagi para pengemudi.

b. Rambu petunjuk

   Rambu yang memberikan petunjuk atau keterangan kepada pengemudi atau pemakai jalan lainnya, tentang arah yang harus ditampuh atau letak kota yang akan dituju lengkap dengan nama dan arah dimana kita berada.

c. Rambu larangan

Rambu ini untuk melarang atau memerintah semua jenis lalu lintas tertentu untuk memakai jalan, jurusan atau tempat-tempat tertentu. Misalnya :

- dilarang berhenti.
- kendaraan harus lewat jalur tertentu.
- semua kendaraan dilarang lewat.

Menurut cara pemasangan dan sifat pesan yang akan disampaikan maka secara garis besar sistem perambuan dapat dikelompokkan atas :

a. Rambu tetap

Rambu tetap adalah semua jenis rambu yang ditetapkan menurut Surat Keputusan Menteri Perhubungan tersebut diatas yang dipasang secara tetap.

b. Rambu tidak tetap

Rambu tidak tetap adalah rambu yang dipasang dan berlaku hanya beberapa waktu, dapat ditempatkan sewaktu-waktu dan dapat dipindah-pindahkan.

Tabel 2.15 Contoh Gambar Rambu-rambu Lalulintas

Nama Rambu-rambu Lalulintas		
Rambu Larangan	**Rambu Peringatan**	**Rambu Petunjuk**
 Dilarang Parkir	 Hati-hati	 Petunjuk Jurusan
 Pejalan Kaki Dilarang Masuk	 Daerah Rawan Longsor	 Tempat Pengisian Bahan Bakar

 Dilarang Mendahului Kendaraan Lain	 Jalan Bergelombang	 Tempat Area Rumah Makan
 Sepeda Motor Dilarang Masuk	 Daerah Banyak Ternak	 Tempat Rumah Sakit

Subanindyo Hadiluwih (2006 : 141 *)*. "Peningkatan jumlah berbagai jenis kendaraan meningkat cukup tajam dari waktu ke waktu, sementara panjang jalan tidak bertambah secara cukup signifikan. Kondisi jalan berikut pengaturan rambu lalu-lintas kurang memadai sehingga terjadi hambatan dan kemacetan lalu-lintas yang seharusnya tidak perlu terjadi. Sikap mental dan disiplin pengguna ".

Dalam catatan kepolisian, kecelakaan yang menyebabkan kematian, tak terlalu banyak di dalam kota. Pada masa penelitian, hal itu tidak terjadi sama sekali. Namun kecelakaan yang menyebabkan matinya orang, terjadi di jalan menuju dan dari luar kota. Baik di Jakarta, Kuala Lumpur maupun di Medan pada umumnya kecelakaan disebabkan oleh kelalaian manusia. Bukan oleh karena kondisi kendaraan maupun kondisi jalan. Mendahului kendaraan lain tanpa memperhitungkan kemungkinan datangnya kendaraan dari depan merupakan kasus tertinggi. Penyebab berikutnya, karena mengemudi dalam keadaan mengantuk. Kasus berikut adalah menabrak kendaraan yang berhenti di pinggir jalan. Kendaraan yang berhenti itupun memang tidak memberikan tanda-tanda yang diwajibkan. Bagaimanapun, faktor manusia *(human error)* sebagai penyebab utama terjadinya kecelakaan amat dominan (94,18%). Penyebab dari faktor lingkungan adalah 4,5% dan faktor kendaraan adalah 1,31%. Adapun yang

dimaksud dengan penyebab faktor manusia antara lain, mengemudi kendaraan terlampau cepat; mengabaikan situasi lalu-lintas; melamun *(lost of memorie)*; ditabrak dan atau menabrak kendaraan lain, seringkali menjadi tabrakan beruntun; berlomba sepeda motor secara tidak resmi di jalan umum *(trek-trekan)*; lelah/mengantuk; mengerem secara mendadak; menerobos lampu merah; dan bertelepon sembari mengemudi.

2.8 SIM (Surat Izin Mengemudi)

Menurut Henry S.Siswosoediro. (2009 : 2) SIM diberikan untuk semua pengemudi kendaraan bermotor, baik kendaraan bermotor roda dua maupun lebih. Untuk mendapatkan SIM ini banyak persyaratan yang harus dipenuhi. Jadi tidak sembarangan orang bisa mendapatkan SIM.

Secara umum, untuk mendapatkan SIM, pemohon akan melalui tiga tahapan penting, antara lain tes kesehatan (terutama mata), ujian teori tentang pengetahuan lalulintas, dan ujian praktik tentang teknik penguasaan kendaraan bermotor, jika semua syarat terpenuhi, pihak kepolisisan akan segera mengeluarkan SIM yang diminta oleh pemohon.

SIM dibagi beberapa golongan diantaranya :

Tabel 2.16 Penggolongan SIM

Nama SIM	Keterangan
SIM A	SIM golongan A digunakan untuk mengemudikan mobil penumpang, mobil bis, dan mobil barang dengan berat yang diperbolehkan tidak lebih dari 3.500 kg.
SIM A Umum	SIM golongan A Umum digunakan untuk mengemudikan mobil atau kendaraan untuk penumpang umum, atau kendaraan penumpang yang memiliki trayek untuk penumpang umum dengan berat yang diperbolehkan tidak lebih dari 3.500 kg.
SIM B1	SIM golongan B1 digunakan untuk mengemudikan

	mobil bis dan mobil barang dengan berat yang diperbolehkan lebih dari 3.500 kg, SIM golongan B1 dapat digunakan sebagai SIM golongan A.
SIM B1 Umum	SIM B 1 Umum digunakan untuk mengemudikan mobil atau kendaraan untuk penumpang umum, atau kendaraan penumpang yang memiliki trayek dengan berat yang diperbolehkan lebih dari 3.500 kg. SIM golongan B1 Umum dapat digunakan sebagai SIM golongan B1 dan A.
SIM B2	SIM golongan B2 digunakan untuk mengemudikan traktor atau kendaraan bermotor dengan menarik kereta tempel atau gandeng dengan berat yag diperbolehkan untuk kereta tempel atau kereta gandeng lebih dari 1.000 kg. SIM golongan B2 dapat digunakan sebagai SIM golongan B1 dan A.
SIM B2 Umum	SIM B2 Umum digunakan untuk mengemudikan kendaraan bermotor yang digunakan untuk penumpang umum atau mobil yang mempunyai trayek untuk penumpang umum yang mempunyai spesifikasi sesuai dengan SIM golongan B2. SIM B2 Umum dapat digunakan sebagai SIM golongan B1, B1 Umum, A, dan A Umum.
SIM C	SIM golongan C digunakan untuk mengemudikan sepeda bermotor yang dirancang mampu mencapai kecepatan lebih dari 40 kilometer per jam. SIM golongan C dapat juga digunakan sebagai SIM golongan D.
SIM D	SIM golongan D digunakan untuk mengemudikan sepeda motor yang dirancang memiliki kecepatan tidak lebih dari 40 kilometer per jam.
SIM Internasional	SIM ini berlaku di semua negara. SIM ini di indonesia diterbitkan oleh Ikatan Motor Indonesia (IMI) yang sudah ditunjuk sebagai instansi penerbit SIM Internasional sesuai dengan golongannya oleh menteri perhubungan dan kepala Polri.
SIM Khusus TNI	SIM ini biasanya digunakan untuk prajurit TNI yang mengendarai kendaraan khusus TNI. Namun SIM untuk anggota TNI ini hanya berlaku pada saat mengemudikan kendaraan bermotor TNI, artinya tidak berlaku untuk mengemudikan kendaraan umum atau kendaraan pribadi.