

**Pemrograman Berorientasi Obyek dengan Java 2 Platform  
Micro Edition (J2ME)**

# Daftar Istilah

<i>Array</i>	: Larik yang berisi elemen-elemen berupa data baik tipe data primitif maupun objek dan mempunyai indeks untuk pengaksesannya
<i>Audio</i>	: Suara
<i>Callback</i>	: Proses penanganan event yang bereaksi setelah event terjadi
<i>Capture</i>	: Proses untuk mengambil potongan/ <i>snapshot</i> dari suatu media dapat audio atau video
<i>Content</i>	: Isi
<i>Event</i>	: Kejadian atau kondisi yang biasanya terjadi karena pengaruh pengguna misalnya karena tombol ditekan
<i>Integer</i>	: Bilangan bulat
<i>Interface</i>	: Kelas yang tidak memiliki implementasi pada method-methodnya (file interface di Java)
<i>Method</i>	: Suatu fungsi atau prosedur yang terdapat dalam suatu kelas
<i>Mobile</i>	: Bergerak
<i>Multimedia</i>	: Penggunaan berbagai media/medium dalam menyampaikan informasi biasanya berupa suara, gambar dan gambar bergerak
<i>Off-screen image</i>	: Gambar/tampilan yang tidak ditampilkan pada <i>display</i>
<i>Overhead</i>	: Kompensasi dari suatu proses yang dijalankan
<i>Pervasif</i>	: Terintegrasi dalam kehidupan sehari-hari ( <i>ubiquitous</i> )
<i>Polling</i>	: Proses penanganan event dimana tidak menunggu suatu event terjadi tapi mengecek komponen tempat event terjadi secara periodik
<i>Protokol</i>	: Bahasa yang dipergunakan untuk berkomunikasi antar perangkat
<i>Provisioning</i>	: Proses menyediakan atau mendistribusikan aplikasi
<i>Record</i>	: Data yang disimpan dalam basis data
<i>Singleton</i>	: Suatu metode perancangan program dimana hanya diijinkan satu obyek untuk keseluruhan program
<i>Socket</i>	: Suatu protokol dimana kedua perangkat mempertahankan <i>state</i> dari komunikasi tersebut dengan membuka port

---

<i>Thread</i>	: Proses yang berdiri sendiri terpisah dari proses utama program dan dapat dieksekusi secara parallel dengan proses utama program
<i>Update</i>	: Proses memperbarui
<i>Video</i>	: Gambar yang bergerak
<i>Vendor</i>	: Perusahaan/produsen

# Daftar Singkatan

API	: <i>Application Program Interface</i>
CDC	: <i>Connected Device Configuration</i>
CLDC	: <i>Connected Limited, Device Configuration</i>
e-Learning	: <i>Electronic Learning</i>
GCF	: <i>Generic Connection Framework</i>
GPRS	: <i>General Packet Radio Service</i>
HTTP	: <i>HyperText Transfer Protocol</i>
HTTPS	: <i>HyperText Transfer Protocol Secure</i>
ICT	: <i>Information Communication Technology</i>
JAD	: <i>Java Application Descriptor</i>
JAR	: <i>Java Archive</i>
J2EE	: <i>Java 2 Platform Enterprise Edition</i>
J2ME	: <i>Java 2 Platform Micro Edition</i>
J2SE	: <i>Java 2 Platform Standard Edition</i>
MIDP	: <i>Mobile Information Device Profile</i>
MMAPI	: <i>Mobile Media Application Program interface</i>
m-Learning	: <i>Mobile Learning</i>
OTA	: <i>Over The Air</i>
URI	: <i>Universal Resource Identifier</i>
URL	: <i>Universal Resource Locator</i>
XML	: <i>Extensible Markup Language</i>
XML-RPC	: <i>Extensible Markup Language Remote Procedure Call</i>

# I Pengenalan Pemrograman Aplikasi Bergerak dan J2ME

---

## I.1 Pengenalan Pemrograman Aplikasi Bergerak

Pemrograman aplikasi bergerak (PAB) tidak banyak berbeda dengan pemrograman konvensional pada PC. Pada pengimplementasiannya perlu diperhatikan aspek karakteristik dari perangkat bergerak itu sendiri yang sering kali mempengaruhi arsitektur dan implementasi dari aplikasi tersebut. Dalam PAB berbagai aspek teknis perangkat implementasi lebih menonjol. Hal ini dikarenakan perangkat bergerak memiliki banyak keterbatasan dibandingkan komputer konvensional atau PC.

Teknologi yang bisa dipergunakan untuk pengimplementasian PAB beragam antara lain WAP, Brew, .Net, i-mode dan J2ME. Masing-masing teknologi ini mempunyai kelebihan dan kekurangan masingmasing. Kelebihan dan kekurangan tersebut dipengaruhi banyak faktor antara lain karakteristik perangkat, kualitas sinyal dan layanan operator serta karakteristik pengguna dari aplikasi bergerak.

Pada buku ini hanya dibahas mengenai pengimplementasian PAB dengan menggunakan J2ME saja. J2ME memiliki beberapa kelebihan yang dirasakan cukup pas diimplementasikan di Indonesia. Kelebihan itu antara lain karakteristik perangkat bergerak/*mobile* di Indonesia lebih mudah mengadopsi teknologi J2ME dibanding dengan teknologi lain.

## I.2 Pengenalan Java dan J2ME

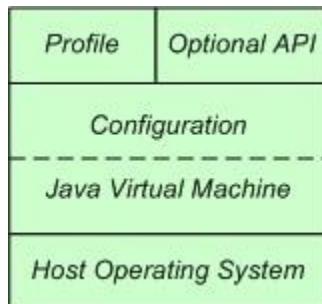
Java merupakan bahasa pemrograman yang diciptakan oleh James Gosling pada tahun 1996 dan mengklaim dirinya mengimplementasikan konsep PBO. Sampai saat ini pengembangan Java berada dibawah Sun Microsystems walaupun akhir-akhir ini Java mulai di *open-soucekan*. Java dapat diimplementasikan pada berbagai aspek kehidupan mulai dari komputer mainframe, PC, telepon genggam/HP, PDA, smart card sampai dengan perlengkapan rumah tangga seperti mesin cuci dan TiVo. Java menjanjikan sifat *platform independent* yang berarti program cukup ditulis satu kali dan *dicompile* satu kali maka akan dapat dijalankan di mesin lain tanpa memerlukan pengubahan kode.

Sampai saat ini Java terbagi menjadi empat kategori yaitu Java 2 Platform Standard Edition (J2SE) untuk aplikasi desktop, Java 2 Platform Enterprise Edition (J2EE) untuk aplikasi server kelas *enterprise* yang biasanya berskala besar, Java 2 Platform Micro Edition (J2ME) untuk aplikasi pada perangkat yang memiliki tingkat komputasi tidak setinggi komputer, misalnya telepon genggam, PDA dan TiVo, dan yang terakhir adalah Java Card yang digunakan untuk pemrograman smart card berbasis Java.



Gambar I-1 Overview teknologi Java

Java 2 Platform Micro Edition (J2ME) dibuat pertama kali oleh Sun Microsystems pada tahun 1998. Tujuan awalnya adalah untuk menyelidiki kemungkinan-kemungkinan dijalankannya Java pada perangkat dengan sumber daya terbatas. J2ME meningkatkan kapabilitas perangkat *mobile* dari yang hanya berkemampuan melakukan komunikasi suara menjadi perangkat yang mampu mengakses internet dan memiliki fungsionalitas lebih dinamis.



Gambar I-2 Arsitektur umum perangkat pendukung J2ME

### I.3 CLDC dan MIDP

J2ME dibagi menjadi dua kategori berdasarkan kapabilitas dari produk-produk tempat diimplementasikannya J2ME. Pembagian kategori ini dilakukan oleh Java Community Process (JCP). Kategori pertama disebut *High-End consumer devices*. Kategori ini memiliki sumber daya yang cukup besar hampir menyamai komputer dalam hal sumber daya listrik, memori maupun bandwidth. Kategori ini diberi label Connected Device Configuration (CDC). Contoh produknya adalah Internet TV.

Kategori kedua disebut *Low-End consumer devices*. Kategori ini memiliki sumber daya yang sangat kecil. Kategori ini diberi nama Connected, Limited Device Configuration (CLDC). Contoh produknya adalah telepon genggam / HP dan two-way pager. Pada modul ini kategori yang dibahas hanya CLDC pada perangkat telepon genggam/HP.

CLDC melingkupi perangkat-perangkat dengan sumber daya terbatas. Sumber daya yang dimaksud disini adalah memori, antarmuka pengguna, daya listrik dan kemampuan prosesor. Karakteristik perangkat CLDC yaitu memori min. 192KB, 16-32 bit prosesor, daya listrik yang rendah dan koneksi jaringan yang tidak stabil.

Ada dua versi CLDC yaitu CLDC 1.0 dan 1.1. Vendor perangkat *mobile* yang menentukan versi mana yang ingin diimplementasikan. Pada CLDC 1.1 dilakukan beberapa upgrade dari versi sebelumnya 1.0 antara lain:

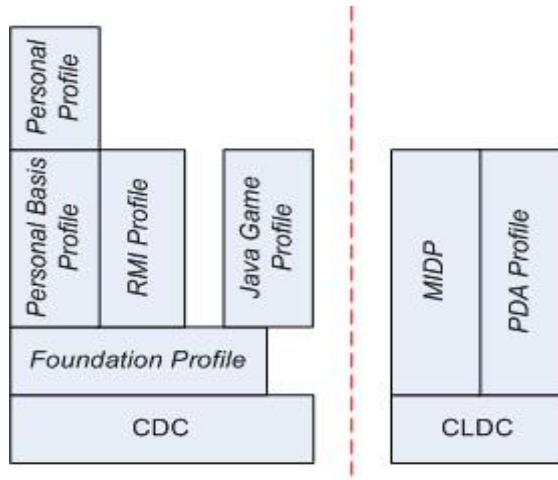
- a. Fitur *floating point*
- b. Persyaratan memori min. 192 KB
- c. Fitur untuk *weak reference*
- d. Fitur detil pada *code verifier*
- e. *Error-handling* yang lebih baik
- f. *Thread naming* dan
- g. *InterruptedException*

CLDC yang diperuntukkan untuk beberapa perangkat membuatnya kesulitan mengeksplorasi kemampuan lebih yang dimiliki. Perangkat yang kurang begitu terfasilitasi dengan standard CLDC ini antara lain telepon genggam/HP. Untuk memberikan kemampuan mengeksplorasi kemampuan terpendam dari perangkat ini, HP, maka Sun Microsystems mengeluarkan satu standard lagi yang diberi nama Mobile Information Device Profile (MIDP).

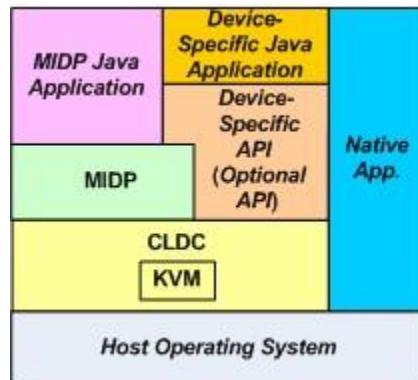
MIDP memungkinkan eksplorasi kemampuan pada perangkat-perangkat CLDC yang berbeda-beda misalnya pada HP. Sampai saat ini ada tiga versi MIDP yang beredar luas dipasaran yaitu versi 1.0, 2.0 dan versi 2.1. Upgrade pada MIDP 2.0 antara lain *Advanced networking*, *Form Enhancement*, *GAME API*, *RGB*

*Images, Code signing dan Permission.* MIDP 2.1 relatif baru dan belum banyak dipergunakan. Meskipun perangkat MIDP masuk dalam kategori CLDC tetapi MIDP mempunyai karakteristik sendiri dibanding perangkat CLDC lainnya yaitu:

- a. Memori (256 KB *non-volatile*, 128KB *volatile*, 8 KB *persistent data*)
- b. Layar tampilan/*Display* (96x54 pix, *Display depth* 1 bit, *Pixel shape ratio* = 1:1)
- c. *Input* (*One/Two handed keyboard*, *touch screen/stylus*)
- d. *Networking* (*Two-way*, nirkabel, *bandwidth* terbatas/*intermittent*)
- e. Multimedia (memiliki kemampuan untuk menjalankan tones)



**Gambar I-3** Arsitektur umum Configuration dan Profile J2ME



**Gambar I-4** Arsitektur implementasi J2ME pada perangkat *mobile*

## I.4 Perangkat Pengembangan J2ME

Kakas/tool untuk mengembangkan aplikasi J2ME disediakan oleh Sun Microsystems. Tool ini diberi nama Wireless ToolKit (WTK). Versi terbaru sampai buku ini ditulis adalah WTK 2.5 untuk Windows sedangkan untuk Unix masih 2.0. Jumlah Java Specification Requirement (JSR) yang diimplementasikan pada masing-

masing versi WTK berbeda-beda. Versi terbaru dari WTK biasanya mengimplementasikan semua JSR yang diimplementasikan pada versi sebelumnya ditambah dengan JSR terbaru lainnya.

Implementasi aplikasi pada WTK ini mengacu pada standar J2ME, CLDC dan MIDP dan tidak mengacu pada perangkat *mobile* vendor manapun. Hal ini perlu diwaspada karena biasanya vendor perangkat *mobile* melakukan penyesuaian implementasi standar pada perangkatnya. Aplikasi yang berjalan baik di WTK Emulator belum tentu dapat berjalan dengan baik di perangkat aslinya.

WTK mempunyai library-library yang dapat digunakan untuk pembuatan aplikasi J2ME. Seringkali penggunaan WTK ini dijadikan plug-in untuk development environment seperti Net Beans dan eclipse. Dengan demikian dimungkinkan pengembangan aplikasi J2ME pada lingkungan yang lebih baik seperti halnya pengembangan aplikasi J2SE maupun J2EE. Library-library yang dicakup dalam WTK antara lain:

- a. Java Technology for Wireless Industry/JTWI (JSR 185)
- b. Wireless Messaging API/WMAPI (JSR 120)
- c. Java API Bluetooth Wireless ToolKit/JABWT (JSR 82)
- d. Java Web Service API (JSR 172)
- e. Mobile Media API (JSR 135)

Selain WTK dari Sun Microsystems terdapat juga perangkat pengembangan yang lain untuk J2ME. Kebanyakan perangkat pengembangan ini dikeluarkan oleh vendor perangkat *mobile*. Dengan menggunakan perangkat pengembangan dari suatu vendor perangkat *mobile* diharapkan nantinya aplikasi kita akan berjalan dengan baik pada perangkat *mobile* vendor tersebut tanpa penyesuaian lagi.

Perangkat-perangkat pengembangan itu antara lain:

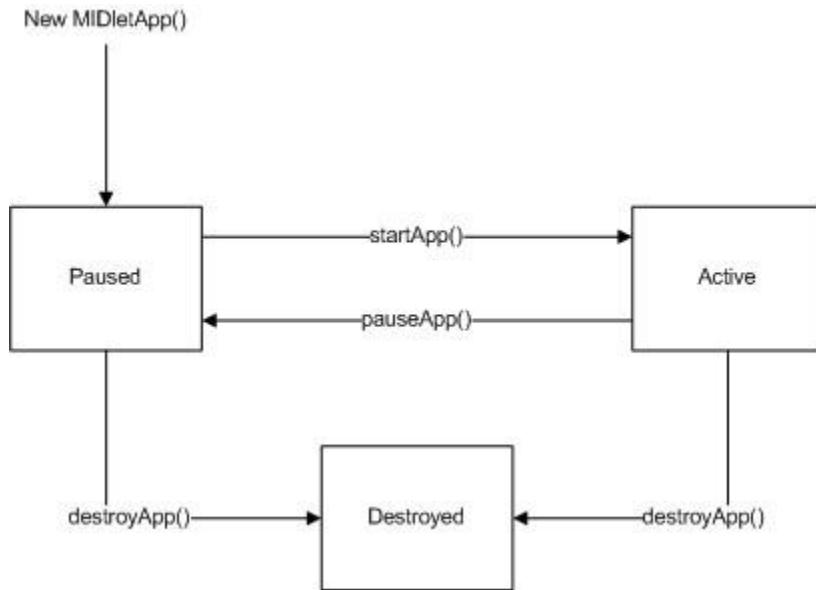
- a. Nokia developer Suite for J2ME
- b. Motorola J2ME SDK
- c. Sony Ericsson J2ME SDK
- d. BenQ-Siemens Mobility Toolkit

## I.5 MIDlet

Aplikasi yang dibuat dengan menggunakan J2ME dengan standar MIDP disebut MIDlet. Kumpulan dari beberapa MIDlet disebut MIDlet *Suite*. Untuk membuat MIDlet diperlukan satu kelas yang menjadi turunan dari kelas `java.microedition.midlet.MIDlet`.

MIDlet mempunyai siklus hidup/life cycle. Suatu MIDlet setelah diciptakan akan berada di salah satu status (*active*, *paused*, *destroyed*). Begitu obyek suatu MIDlet diciptakan akan memasuki status *paused* menunggu perintah berikutnya. MIDlet memasuki status *active* ketika setelah *method startApp ()*

dipanggil. MIDlet akan kembali ke status pause jika *method* `pauseApp()` dipanggil. Setelah semua proses di dalam MIDlet selesai dilakukan dan MIDlet dimatikan maka MIDlet berada dalam status *destroyed*. Status *destroyed* ini dimasuki MIDlet pada pemanggilan *method* `destroyApp(boolean)`.



**Gambar I-5** Siklus hidup (*Life Cycle*) MIDlet

## I.6 Aplikasi J2ME Pertama

Pada sub-bab ini akan kita tunjukkan salah satu aplikasi J2ME yang sangat sederhana. Meneruskan tradisi belajar bahasa pemrograman, kita akan membuat program yang menampilkan tulisan Hello World dengan J2ME. Berikut akan ditampilkan kode dari aplikasi Hello World beserta tampilannya dalam emulator WTK. Selanjutnya akan diberikan penjelasan mengenai kode program Hello World

### Listing I-1 MIDlet HelloWorld

---

```

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloWorldMidlet extends MIDlet implements CommandListener {
private Command cmdExit;      public HelloWorldMidlet(){
    cmdExit      = new Command("Exit", Command.EXIT, 1);
    TextBox tb = new TextBox("Hello MIDlet", "Hello, World!", 15, 0);
    tb.addCommand(cmdExit);          tb.setCommandListener(this);
    Display.getDisplay(this).setCurrent(tb);
}
protected void startApp() {}
protected void pauseApp() {}
protected void destroyApp(boolean force) {}      public
void commandAction(Command c, Displayable d) {
    if (c == cmdExit) notifyDestroyed();
}
  
```

```
    }  
}
```

---



**Gambar I-6** Tampilan aplikasi Hello World

Pada aplikasi diatas kita gunakan perangkat pengembangan WTK dari Sun Microsystems. Aplikasi Hello World sangat sederhana sehingga dapat di install di berbagai perangkat *mobile* tanpa ada masalah yg berarti dan tanpa penyesuaian. Hal ini kemungkinan tidak berlaku jika aplikasi J2ME yang dibuat adalah aplikasi besar, kompleks dan menggunakan fitur-fitur tertentu.

Aplikasi Hello World terdiri dari satu kelas yang bernama `HelloWorld`. Karena kita bertujuan membuat MIDlet maka kelas tersebut harus merupakan turunan dari `javax.microedition.midlet.MIDlet`. `HelloWorld` harus mengimplementasikan tiga *method* dari orang tuanya yaitu `startApp()`, `pauseApp()` dan `destroyApp()`. Untuk menangani interaksi dengan pengguna aplikasi ini dilengkapi dengan satu obyek `Command` bernama `exitCommand`. `HelloWorld` juga mengimplementasikan interface bernama `CommandListener`. `CommandListener` ini berisi satu *method* bernama `commandAction()` yang berisi perintah respon jika suatu obyek `Command` dieksekusi dalam hal ini obyek `Command` tersebut adalah `exitCommand` dan responnya adalah untuk menutup aplikasi MIDlet Hello World.

Bila diperhatikan *method* `startApp()` yang merupakan entry point dari semua aplikasi MIDlet tidak berisi kode. Hal ini dikarenakan secara otomatis konstruktor `HelloWorld` pasti dipanggil pertama kali dan di dalamnya sudah terdapat kode untuk mengatur obyek-obyek yang diperlukan dan menampilkannya.

## II Antarmuka Tingkat Atas

---

### II.1 Antarmuka di J2ME

J2ME, dalam MIDP, seperti halnya kategori java lainnya (J2SE) mendefinisikan serangkaian library untuk membuat tampilan antarmuka pengguna/user interface. Pada MIDP terdapat dua jenis library antarmuka pengguna yaitu:

- a. Antarmuka tingkat atas/*High-level User Interface*
- b. Antarmuka tingkat bawah/*Low-level interface*

Antarmuka tingkat atas terdiri dari komponen-komponen yang telah terdefinisi/*well-defined* dan siap langsung digunakan dalam MIDlet. Komponen ini mampu merespon input dari pengguna langsung tanpa penambahan kode. Semua komponen-komponen yang berada pada tingkat atas ini merupakan turunan dari sebuah kelas yang bernama *Screen*. Pada bab ini khusus dibahas antarmuka tingkat atas ini.

Antarmuka tingkat bawah menyediakan kendali lebih dalam pembentukan komponen baik tampilan maupun interaksinya. Antarmuka jenis ini biasanya digunakan untuk pembuatan antarmuka pengguna grafis/*graphical user interface* pada aplikasi game maupun aplikasi-aplikasi lain yang memerlukan komponen antarmuka yang tidak terdapat pada antarmuka pengguna tingkat atas. Antarmuka tingkat ini direpresentasikan oleh *Canvas* dan turunannya yaitu *GameCanvas*. Pembahasan mengenai antarmuka tingkat bawah ini ada pada Bab III.

### II.2 Display, Displayable dan Ticker

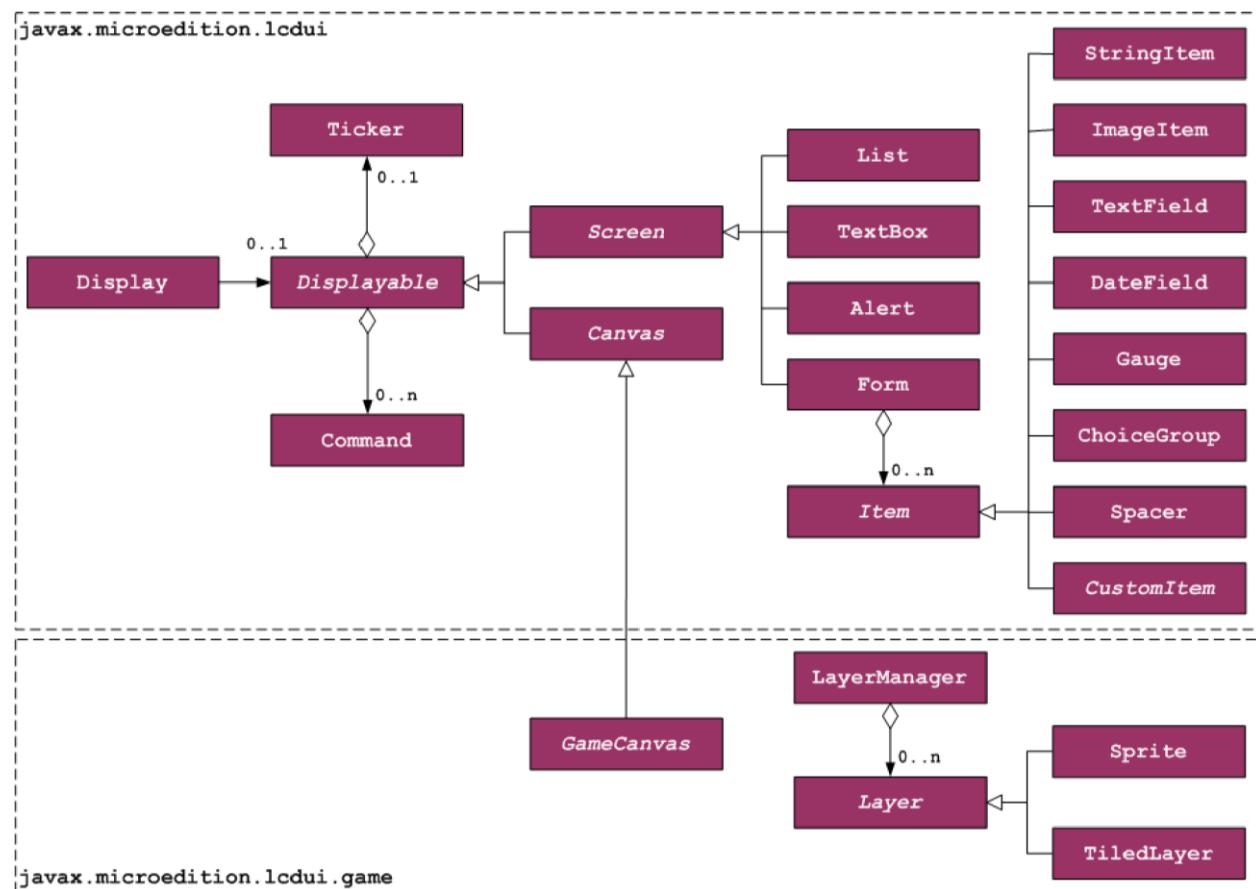
*Display* adalah representasi dari layar tampilan perangkat bergerak. *Display* ini tidak dapat diciptakan dan hanya ada satu (*singleton*). Untuk mendapatkan obyek ini dapat digunakan *static method* dari *Display* yaitu sebagai berikut.

```
static Display getDisplay(Midlet m)
```

Obyek yang dapat ditampilkan melalui obyek *Display* ini adalah obyek *Displayable* dan turunannya serta obyek *Alert*. Jika ingin menampilkan *Alert* maka obyek *Displayable* yang akan ditampilkan selanjutnya juga harus didefinisikan. Untuk menampilkan obyek pada layar perangkat digunakan *method* sebagai berikut

- a. void setCurrent(Displayable nextDisplayable)
- b. void setCurrent(Alert alert, Displayable nextDisplayable)

Obyek `Displayable` adalah sebuah obyek yang dapat tampil pada obyek `Display` sendirian tanpa kehadiran obyek lagi. Turunan `Displayable` ini ada dua yaitu `Screen` dan `Canvas`. `Screen` dan turunannya masuk ke dalam antarmuka tingkat atas, dibahas di bab ini, sedangkan `Canvas` dan turunannya masuk dalam antarmuka tingkat bawah, dibahas pada bab berikutnya.



**Gambar II-1** Hirarki kelas antarmuka (tingkat atas dan bawah) pada J2ME

`Ticker` adalah sebuah obyek yang unik dan dapat tampil baik di antarmuka tingkat atas maupun antarmuka tingkat bawah. `Ticker` berfungsi menampilkan teks bergerak pada sebuah obyek turunan `Displayable` baik itu `Screen` maupun `Canvas`. Biasanya posisi `Ticker` dilayar berada pada posisi paling atas walaupun mungkin saja vendor meletakkannya ini pada posisi lain selain paling atas.

`Ticker` tidak dapat diatur kapan mulainya, kapan berhentinya maupun diatur kecepatan geraknya. Untuk memasukkan `Ticker` ini digunakan *method* sebagai berikut.

```
public void setTicker(Ticker)
```

#### **Listing II-1** Contoh sebuah `List` dan `Ticker`

---

```
Ticker ticker = new Ticker("Career Days in ITB this January. be There!!!");
listJobs = new List("Vacancies",Choice.IMPLICIT);
```

```
listJobs.append("Programmer", null);listJobs.setTicker(ticker);
listJobs.append("System Analyst", null);
listJobs.append("GUI Designer", null);
listJobs.append("Network Administrator", null);
listJobs.append("Database Administrator", null);
listJobs.append("Technical Suport", null);
listJobs.setCommandListener(this);
display.setCurrent(listJobs);
```

---



**Gambar II-2** Sebuah obyek `Displayable` (List) dengan Ticker

### II.3 Item dan Form

Item adalah komponen antarmuka tingkat atas yang bukan merupakan turunan `Displayable`. Item ini mempunyai sifat tidak dapat berdiri sendiri tanpa kehadiran komponen antarmuka tingkat tinggi lain yaitu Form. Agar Item dapat digunakan, obyek Item harus berada di dalam Form. Secara teknis yang dimasukkan ke dalam Form bukan obyek/*instance* dari Item tapi obyek/*instance* dari kelas turunan Item. Kelas-kelas turunan Item adalah `StringItem`, `ImageItem`, `TextField`, `Datefield`, `Gauge`, `ChoiceGroup`, `Spacer` dan `CustomItem`. Masing-masing turunan Item ini memiliki atribut ukuran dan *layout* masing-masing.

Form adalah komponen turunan `Screen` yang berfungsi sebagai tempat untuk menampung (*container*) bagi komponen lain. Fungsi Form pada J2ME sama seperti Form pada HTML. Form selain memiliki obyek Item biasanya juga mempunyai obyek Command. Fungsi Command pada Form ini juga sama dengan fungsi Button pada Form di HTML. Command berfungsi sebagai *trigger* untuk memulai suatu proses. Form juga dapat memasukkan String ataupun Image. Dalam hal ini obyek-obyek tersebut dibungkus menjadi Obyek turunan Item terlebih dahulu. Mengenai hal ini akan dibahas pada sub bab tentang `StringItem` dan `ImageItem`. Untuk memasukkan suatu obyek Item ke dalam

Form dapat digunakan beberapa *method* sebagai berikut:

- a. void insert(int itemNum, Item item)
- b. int append(Image img)
- c. int append(Item item)
- d. int append(String str)

*Method* `insert()` digunakan jika kita ingin memasukkan suatu `Item` pada posisi tertentu dalam `Form`. Posisi ini ditentukan dengan biangan indeks. Sedangkan *method* `append()` digunakan untuk memasukkan `Item` pada posisi paling akhir/bawah pada `Form`. *Method* `append()` mengembalikan posisi dari suatu `Item` dalam bilangan indeks.

Sedangkan untuk menghapus obyek `Item` dari `Form` dapat digunakan *method* `delete(int)` atau `deleteAll()` untuk menghapus semua obyek `Item` di dalam `Form`.

`Form` mempunyai sifat seperti `Collection`. Obyek-obyek `Item` yang tersimpan di dalamnya dapat diakses dengan menggunakan index, *method* yang digunakan adalah `get(int)`, dan dapat diketahui jumlah total obyek `Item` didalamnya dengan menggunakan *method* `size()`. Index obyek `Item` dalam `Form` dimulai dari 0 sehingga index maksimum adalah `size() - 1`.

Tidak ada batasan berapa jumlah minimum dan maksimum obyek `Item` yang dapat dimasukkan ke `Form`. Akan tetapi `Form` sebaiknya dibuat sependek dan sesederhana mungkin karena pada umumnya pengguna perangkat bergerak tidak menyukai *scrolling* dan `Form` yang terlalu kompleks.

## II.4 TextField

`TextField` adalah komponen turunan `Item` yang menampung teks dan mengedit teks tersebut. `TextField` mempunyai ukuran maksimum, label dan format input. Beberapa vendor mengimplementasikan `TextField` sehingga ukuran `TextField` dapat secara otomatis membesar sesuai dengan input string yang dimasukkan. Berapa jumlah baris dan kolom yang ditampilkan `TextField` menjadi tanggung jawab dari vendor perangkat bergerak dalam pengimplementasiannya. Ada komponen lain selain `TextField` di antarmuka tingkat atas yang mempunyai fungsi sama dengan `TextField` yaitu `TextBox`. `TextBox` merupakan turunan `Screen` dan buka turunan `Item`. Perbedaan antara `TextField` dan `TextBox` terletak pada jumlah maksimum karakter, multi-line dan format input.

Masukan/input pada `TextField` dapat di format sesuai kebutuhan. Ada enam jenis format yang dikenali oleh `TextField` yaitu ANY, EMAILADDR, NUMERIC, PHONENUMBER, URL dan DECIMAL. Semua format input diatas merupakan konstanta static dari class `TextField`. Untuk format EMAILADDR, PHONENUMBER dan URL dapat mendapatkan perlakuan khusus tergantung implementasi vendor. Perlakukan khusus ini antara lain kelebihan untuk melakukan lookup atau melihat ke dalam *phonebook* perangkat *mobile* untuk memasukkan nomor telpon, email ataupun alamat URL.

**Tabel II-1** Format `TextField`

Format	Keterangan
TextField.ANY	Pengguna dapat memasukkan karakter/input apapun kedalam TextField
TextField.EMAILADDR	Hanya mengijinkan input berupa alamat email tetapi implementasi vendor memungkinkan akses untuk <i>lookup</i> pada <i>phonebook</i> perangkat <i>mobile</i>
TextField.NUMERIC	Hanya mengijinkan input berupa bilangan bulat
TextField.URL	Hanya mengijinkan input berupa alamat URL (Universal Resource Locator)
TextField.DECIMAL	Memungkinkan input berupa bilangan desimal/pecahan
TextField.PHONENUMBER	Memungkinkan masukan berupa nomor telpon. Implementasi vendor dapat saja membuat format ini sama dengan format NUMERIC dan mungkin juga mendapatkan akses untuk <i>lookup</i> pada <i>phonebook</i> perangkat <i>mobile</i>

Dalam TextField juga dapat didefinisikan perilaku/sifat TextField setiap kali user memasukkan input/karakter. Jenis-jenisnya antara lain PASSWORD, UNEDITABLE, SENSITIVE, NON\_PREDICTIVE, INITIAL\_CAPS\_WORD dan INITIAL\_CAPS\_SENTENCE. Suatu obyek TextField dapat memiliki salah satu atau lebih dari perilaku ini. Jika hal ini dilakukan perlu diperhatikan kompatibilitas antar perilaku selain itu juga implementasi antar perangkat bergerak.

**Tabel II-2** Perilaku TextField terhadap masukan User

Perilaku	Keterangan
TextField.PASSWORD	Masukan karakter disamarkan dengan karakter khusus misalnya asteriks (*)
TextField.UNEDITABLE	Teks yang terdapat pada TextField tidak diperbolehkan untuk diubah
TextField.SENSITIVE	Teks yang dimasukkan ke dalam TextField bersifat rahasia dan tidak boleh disimpan di memori
TextField.NON_PREDICTIVE	Teks yang dimasukkan tidak terdapat dalam kamus sehingga tidak cocok bila digunakan prediksi input
TextField.INITIAL_CAPS_WORD	Huruf pertama dari masing-masing kata pada teks masuk dari user harus menjadi huruf besar
TextField.INITIAL_CAPS_SENTENCE	Huruf pertama dari keseluruhan teks yang dimasukkan oleh user diganti menjadi huruf besar

Berikut akan ditunjukkan contoh penggunaan TextField dalam membuat sebuah Form sederhana untuk login. Pada contoh dibawah terdapat dua buah obyek TextField yang satu tempat memasukkan

user name dan yang kedua password. TextField password menggunakan atribut TextField.PASSWORD.

#### **Listing II-2** Contoh penggunaan TextField

```
public class LoginForm extends Form { private TextField  
txtUserName, txtPassword, txtServer; private Command  
cmdLogin, cmdExit, cmdClear, cmdRegister; public  
LoginForm(){ super("Simple Login Form");  
    txtUserName=new TextField("username","",255,TextField.EMAILADDR);  
    txtPassword=new TextField("password","",255, TextField.PASSWORD);  
    cmdClear      = new Command("Clear", Command.SCREEN,2);  
    cmdLogin     = new Command("Login",Command.SCREEN,1);  
    cmdExit      = new Command("Exit",Command.EXIT,2);  
    cmdRegister  = new Command("Register",Command.SCREEN,3);  
    addCommand(cmdExit); addCommand(cmdClear);  
    addCommand(cmdLogin); addCommand(cmdRegister);  
    append(txtUserName); append(txtPassword);  
    append(txtServer); }  
}
```



**Gambar II-3** Form Sederhana untuk Login

#### **II.5 StringItem**

StringItem merupakan komponen untuk menampilkan tulisan pada layar. StringItem ini membungkus obyek String supaya dapat diperlakukan sama dengan turunan obyek Item yang lainnya. Obyek String sendiri dapat langsung dimasukkan ke dalam Form dengan method int append(String) tapi yang terjadi sebenarnya adalah obyek tersebut dibungkus dengan menggunakan StringItem. Jika beberapa obyek StringItem dimasukkan ke dalam Form secara berurutan maka semua obyek tersebut akan di gabungkan/konkatenasi.

Ada tiga jenis tampilan dari StringItem yaitu PLAIN, BUTTON dan HYPERLINK. Secara *default* suatu obyek StringItem memiliki penampilan PLAIN. Jika menggunakan jenis BUTTON atau HYPERLINK maka perlu didefinisikan juga obyek Command yang akan merespon user jika menekan StringItem tersebut. Perbedaan antara BUTTON dan HYPERLINK terletak pada tampilannya. Dengan menggunakan BUTTON akan tiampilkan StringItem dengan tampilan seperti tombol sedangkan HYPERLINK akan menampilkan StringItem seperti link pada web. Obyek Command yang didefinisikan diatas dapat menjadi aksesibel pada keseluruhan Form ataupun hanya pada StringItem yang memakainya.

**Listing II-3** Contoh penggunaan StringItem

---

```
import javax.microedition.lcdui.Command; import
javax.microedition.lcdui.CommandListener; import
javax.microedition.lcdui.Displayable; import
javax.microedition.lcdui.Form; import
javax.microedition.lcdui.Item; import
javax.microedition.lcdui.StringItem; import
javax.microedition.lcdui.TextField;

public class LoginForm extends Form { private TextField
    txtUserName, txtPassword, txtServer; private Command
    cmdLogin, cmdExit, cmdClear, cmdRegister; public
    LoginForm() {
        super("Simple Login Form");
        txtUserName = new TextField("username","",255, TextField.EMAILADDR);
        txtPassword = new TextField("password","",255, TextField.PASSWORD);
        txtServer   = new TextField("server","",255, TextField.HYPERLINK);
        cmdClear    = new Command("Clear", Command.SCREEN,2);
        cmdLogin   = new Command("Login",Command.SCREEN,1);
        cmdExit    = new Command("Exit",Command.EXIT,2);
        cmdRegister = new Command("Register",Command.SCREEN,3);
        Item forgetPwd = new StringItem(null,"Forget Your Password? Click");
        Item link      = new StringItem(null,"Here", Item.HYPERLINK);
        Command cmdHelp = new Command("Help Password",Command.ITEM,1);
        link.setDefaultCommand(cmdHelp); addCommand(cmdExit);
        addCommand(cmdClear); addCommand(cmdLogin);
        addCommand(cmdRegister); addCommand(cmdHelp);
        append(txtUserName); append(txtPassword); append(txtServer);
        append(forgetPwd); append(link);
    }
}
```

---



**Gambar II-4** StringItem pada Form

## II.6 ImageItem

ImageItem seperti halnya StringItem merupakan pembungkus untuk obyek lain. Dalam hal ini adalah Image. Gambar dapat dimasukkan ke dalam Form dalam bentuk Image langsung maupun dibungkus dengan menggunakan ImageItem. Konsep penambahan Image pada Form juga sama seperti penambahan String pada Form. Secara *default* ImageItem akan ditempatkan di sebelah StringItem atau ImageItem yang telah berada sebelumnya. Jika tempat untuk ImageItem tersebut tidak cukup maka akan dimasukkan kedalam baris baru.

ImageItem selain mempunyai layout warisan dari Item, juga mempunyai layout sendiri untuk penempatkannya didalam Form. Layout-layout itu yaitu LAYOUT\_DEFAULT, LAYOUT\_CENTER, LAYOUT\_RIGHT, LAYOUT\_LEFT, LAYOUT\_NEWLINE\_BEFORE dan LAYOUT\_NEWLINE\_AFTER. Layout ini mengatur letak penempatan ImageItem pada Form atau layar perangkat. Penentuan layout ini dilakukan pada saat konstruksi obyek ImageItem atau dengan *method* sebagai berikut.

```
void setLayout(int layout)
```

Meskipun pada spesifikasi J2ME terdapat layout yang dapat digunakan oleh ImageItem pada implementasinya tergantung pada vendor perangkat perangkat *mobile*. Contoh mengenai ImageItem dapat dilihat pada sub-bab Contoh Aplikasi MIDlet Menggunakan Antarmuka Tingkat Atas.

## II.7 DateField

DateField merupakan komponen untuk memilih dan menampilkan tanggal atau waktu pada antarmuka J2ME. Tampilan komponen ini berbeda-beda untuk masing-masing perangkat. DateField

dapat diatur apakah digunakan untuk menampilkan informasi tanggal, waktu atau keduanya dengan memilih tipe DATE, TIME atau DATE\_TIME pada saat pembentukan obyek DateField.

Pembentukan tanggal atau waktu pada DateField dapat menggunakan obyek dari java.util.Date.

```
void setDate(Date date)
```

Contoh mengenai DateField dapat dilihat pada sub-bab Contoh Aplikasi MIDlet Menggunakan Antarmuka Tingkat Atas.

## II.8 ChoiceGroup

ChoiceGroup menampilkan daftar elemen yang dapat dipilih di dalam Form seperti halnya List. ChoiceGroup memungkinkan memilih satu atau lebih dari satu elemen yang terdapat dalam daftarnya. Sama seperti List, ChoiceGroup juga memungkinkan setiap elemen yang ditampilkan berasosiasi dengan gambar tertentu. Sering kali ChoiceGroup ini diasosiasikan dengan RadioButton atau CheckBox. Untuk memasukkan suatu elemen ke dalam ChoiceGroup digunakan *method* sebagai berikut.

```
int append(String stringPart, Image imagePart)
```

ChoiceGroup memungkinkan mode POPUP dimana hanya satu elemen yaitu elemen yang telah dipilih saja yang ditampilkan dilayar. Elemen lain akan ditampilkan jika user ingin memilih elemen lain. POPUP ini hanya memungkinkan pemilihan satu elemen. Implementasi ChoiceGroup juga tergantung pada vendor perangkat.

Berikut akan diberikan contoh penggunaan ChoiceGroup untuk melengkapi form lamaran pekerjaan.

Akan ditampilkan dua jenis ChoiceGroup yaitu yang EXPLICIT dan MULTIPLE.

### Listing II-4 Contoh penggunaan ChoiceGroup

---

```
ChoiceGroup degreeGroup = new ChoiceGroup("Degree:", Choice.EXCLUSIVE);
degreeGroup.append("Bachelor", null); degreeGroup.append("Master", null);
degreeGroup.append("Doctor", null);
ChoiceGroup positionGroup = new ChoiceGroup("Position:", Choice.MULTIPLE);
positionGroup.append("Engineer", null); positionGroup.append("IT", null);
positionGroup.append("Network", null); positionGroup.append("Accounting", null);
Form f = new Form("Application Form");
f.append(degreeGroup);
f.append(positionGroup);
```

---



Gambar II-5 Tampilan ChoiceGroup

## II.9 Gauge

Gauge merupakan komponen yang fungsinya sama dengan ProgressBar pada J2SE. Gauge merepresentasikan nilai dari 0 sampai bilangan yang dapat didefinisikan sendiri. Gauge dapat bersifat interaktif dimana user dapat berinteraksi dengan Gauge (menggunakan ItemStateListener) ataupun non-interaktif dimana Gauge hanya berfungsi seperti ProgressBar biasa. Suatu obyek Gauge ditentukan pada saat Gauge dibuat (ditentukan saat pemanggilan konstruktor). Setelah suatu Gauge di set interaktif atau non-interaktif maka sifatnya tidak dapat diubah lagi.

```
Gauge (String label, boolean interactive, int maxValue,  
int initialValue)
```

Nilai dari Gauge dapat diakses oleh aplikasi MIDlet baik Gauge itu interaktif ataupun non-interaktif. Untuk Gauge non-interaktif nilai maksimum dapat di set pada nilai tak terhingga. Terdapat konstanta INDEFINITIVE yang dapat digunakan untuk mengatur nilai tak terhingga. Konstanta ini digunakan sebagai salah argumen pada konstruktor atau argumen pada *method* setMaxValue(int).

```
void setMaxValue(int maxValue)
```

Untuk Gauge dengan nilai maksimum tak terhingga ada dua mode yang dapat digunakan yaitu CONTINUOUS\_RUNNING dan INCREMENTAL\_UPDATING. CONTINUOUS\_RUNNING secara otomatis akan melakukan update tampilan Gauge (nilainya bertambah) sedangkan pada INCREMENTAL\_UPDATING menunggu *method* setValue(int) dipanggil dengan argumen Gauge.INCREMENTAL\_UPDATING. Untuk masing-masing mode ini terdapat konstanta yang mengindikasikan tidak ada aktivitas Gauge yaitu CONTINUOUS\_IDLE dan INCREMENTAL\_IDLE. Implementasi Gauge non-interaktif dengan nilai maksimum tak terhingga dapat berupa gambar animasi

yang bergerak-gerak misalnya pada WTK direpresentasikan Duke yang bergerak.Untuk Gauge yang bersifat interaktif, nilai dari Gauge dapat diatur dengan menggunakan *method* sebagai berikut.

```
void setValue(int value)
```

Untuk Gauge yang bersifat non-interaktif pengaturan nilai Gauge menggunakan *method* diatas tidak diperbolehkan.

---

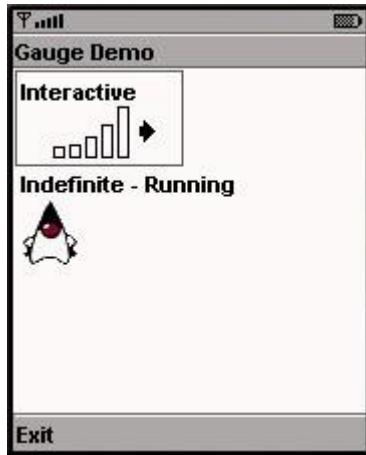
**Listing II-5 File GaugeDemo.java**

```
import javax.microedition.lcdui.Command; import
javax.microedition.lcdui.CommandListener; import
javax.microedition.lcdui.Display; import
javax.microedition.lcdui.Displayable; import
javax.microedition.lcdui.Form; import
javax.microedition.lcdui.Gauge; import
javax.microedition.midlet;

public class GaugeDemo extends MIDlet implements CommandListener {
    private Command cmdExit;    private Display display;    private Form
mainForm;    public GaugeDemo() {
    display = Display.getDisplay(this);
    cmdExit = new Command("Exit", Command.EXIT, 1);
    mainForm = new Form("Gauge Demo");
    Gauge interactiveGauge = new Gauge("Interactive", true, 10, 0);
    Gauge nonInteractiveGauge = new Gauge("Indefinite - Running", false,
                                         Gauge.INDEFINITE, Gauge.CONTINUOUS_RUNNING);
    mainForm.append(interactiveGauge);
    mainForm.append(nonInteractiveGauge);           mainForm.addCommand(cmdExit);
    mainForm.setCommandListener(this);
}
protected void startApp(){display.setCurrent(mainForm);}
protected void destroyApp(boolean unconditional) {}
protected void pauseApp() {}

public void commandAction(Command c, Displayable d) {
destroyApp(false);           notifyDestroyed();
}
}
```

---



**Gambar II-6** Gauge interaktif dan non-interaktif pada WTK

## II.10 CustomItem

CustomItem memungkinkan terbentuknya komponen baru yang dapat dimasukkan ke dalam Form. CustomItem dapat menerima respon dari user menggunakan ItemStateListener.

CustomItem adalah kelas *abstract* karena itu untuk membuat komponen baru kita perlu membentuk kelas yang merupakan turunan dari CustomItem. CustomItem mempunyai sifat-sifat yang mirip dengan Canvas, akan diterangkan di bab berikutnya, tetapi mempunyai kelebihan dapat dimasukkan ke dalam Form.

Kelas yang merupakan turunan CustomItem harus mengimplementasi lima *method* yaitu:

- a. protected int getMinContentWidth()
- b. protected int getMinContentHeight()
- c. protected int getPrefContentWidth()
- d. protected int getPrefContentHeight()
- e. protected abstract void paint(Graphics g, int w, int h)

Perbedaan *method* paint() di CustomItem dan paint() pada Canvas adalah pada CustomItem *method* paint() memiliki tambahan argumen panjang dan lebar area yang akan dibentuk. Hal ini dikarenakan komponen CustomItem tidak menempati seluruh layar seperti pada Canvas. *Method* repaint() juga tersedia pada CustomItem sama seperti pada Canvas().

*Method* repaint() ini bukan *method* *abstract* sehingga implementasinya pada kelas turunan menjadi opsional. *Method* repaint() pada CustomItem ada dua yaitu:

- a. protected void repaint()
- b. protected void repaint(int x, int y, int w, int h)

Selain *method* `paint()` dan `repaint()` ada satu *method* lagi yang seringkali digunakan pada saat menurunkan kelas `CustomItem` yaitu *method* `traverse()`. *Signature* dari *method* `traverse()` adalah sebagai berikut:

```
protected boolean traverse(int dir, int viewPortWidth, int  
viewPortHeight, int[] visRect_inout)
```

*Method* ini memungkinkan pergerakan *pointer* didalam obyek turunan `CustomItem` (*traversing*). *Method* ini memungkinkan proses *scrolling* dalam obyek turunan `CustomItem` sehingga dapat pindah-pindah area dalam obyek `CustomItem`. Contoh penggunaannya jika kita ingin membuat tabel dari `CustomItem` dan kita ingin berpindah-pindah antar sel dalam tabel tersebut. Untuk mengetahui apakah suatu perangkat *mobile* mampu melakukan *traversing* secara horisontal atau vertikal digunakan *method* sebagai berikut:

```
protected int getInteractionModes()
```

Ketika proses *traversing* mulai dilakukan atau berakhir, `CustomItem` (dan turunannya otomatis memanggil *method* sebagai berikut:

```
protected void traverseOut()
```

*Method* diatas dapat di override dengan menambahkan perilaku khusus ketika proses *traversing* dimulai ataupun akan berakhir. Setiap turunan `CustomItem` harus mengetahui jenis interaksi apa saja yang diimplementasikan pada perangkat mobile. Jenis-jenis interaksi yang mungkin dimiliki oleh perangkat antara lain `KEY_PRESS`, `KEY_RELEASE`, `KEY_REPEAT`, `POINTER_PRESS`, `POINTER_RELEASE`, `POINTER_DRAG`, `TRAVERSE_HORIZONTAL` dan `TRAVERSE_VERTICAL`. Suatu obyek `CustomItem` dapat saja berinteraksi dengan menggunakan lebih dari satu cara tergantung perangkat yang digunakan. Untuk itu implementasi obyek turunan `CustomItem` perlu menentukan cara interaksi mana saja yang akan dipakai.

Berikut akan ditunjukkan sebuah kelas yang merupakan turunan dari `CustomItem`. Kelas tersebut menampilkan tabel pada layar perangkat *mobile*. Kelas ini nanti akan memungkinkan *traversing* mengakses masing-masing sel dalam tabel.

---

**Listing II-6** Kelas `Table.java`

```
import javax.microedition.lcdui.*;  
  
public class Table  
extends CustomItem  
implements ItemCommandListener {  
  
    private final static Command CMD_EDIT = new Command("Edit",
```





```

        repaint(currentX * dx, (currentY + 1) * dy, dx,
                currentX = cols - 1;
                repaint(currentX * dx, currentY * dy, dx, dy);
            } else {
        location = UPPER;
        return false;
    }
}
case Canvas.DOWN:           case
Canvas.RIGHT:               if
(location == UPPER) {       } else {
location = IN;             if (currentX < (cols - 1)) {
dy);                      currentX++;
repaint((currentX - 1) * dx, currentY * dy, dx,
repaint(currentX * dx, currentY * dy, dx, dy);
} else if (currentY < (rows - 1)) {
currentY++;
repaint(currentX * dx, (currentY - 1) * dy, dx,
currentX = 0;
repaint(currentX * dx, currentY * dy, dx, dy);
} else {
location = LOWER;
return false;
}
}
visRect_inout[0] = currentX;
visRect_inout[1] = currentY;
visRect_inout[2] = dx;
visRect_inout[3] = dy;

return true;
}
public void setText(String text) {
data[currentY][currentX] = text;
repaint(currentY * dx, currentX * dy, dx, dy);
}

public void commandAction(Command c, Item i) {
if (c == CMD_EDIT) {
TextInput textInput = new TextInput(data[currentY][currentX],
this, display);           display.setCurrent(textInput);
}
}
}

```

---

**Listing II-7 File** TableDemo.java

---

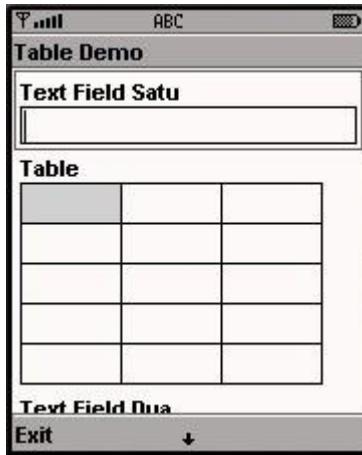
```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.MIDlet;

public class CustomItemDemo extends MIDlet implements CommandListener {
    private final static Command CMD_EXIT = new Command("Exit",
        Command.EXIT, 1);
    private Display display;

    private boolean firstTime;
    private Form mainForm;

    public CustomItemDemo() {
        firstTime = true;
        mainForm = new Form("Table Demo");
    }
    protected void startApp() {
        if (firstTime) {
            display = Display.getDisplay(this);
            mainForm.append(new TextField("Text Field Satu", null, 10,
                0));
            mainForm.append(new Table("Table",
                Display.getDisplay(this)));
            mainForm.append(new TextField("Text Field Dua", null, 10,
                0));
        }
        mainForm.addCommand(CMD_EXIT);
        mainForm.setCommandListener(this);
        firstTime = false;
    }
    display.setCurrent(mainForm);
}
    public void commandAction(Command c, Displayable d) {
        if (c == CMD_EXIT) { destroyApp(false);
            notifyDestroyed();
        }
    }
    protected void destroyApp(boolean unconditional) {}
protected void pauseApp() {}
}
```

---



**Gambar II-7** Tampilan Tabel turunan dari CustomItem

## II.11 List

List fungsinya adalah menampilkan daftar pilihan pada layar. Masing-masing elemen yang ditampilkan direpresentasikan oleh String dan dapat pula ditambahkan dengan gambar/Image. Setiap elemen yang ditampilkan dalam List dapat dipilih dan List akan bereaksi terhadap pilihan user karena memiliki CommandListener. Untuk memasukkan elemen ke dalam List kita dapat menggunakan beberapa *method* yaitu:

- a. void insert(int elementNum, String stringPart, Image imagePart)
- b. int append (String stringPart, Image imagePart)

Jika menggunakan *method* pertama kita dapat memasukkan elemen pada urutan/index tertentu dalam List sedangkan *method* kedua akan otomatis menambahkan eleman pada bagian akhir. Untuk menghapus elemen dari List digunakan *method* sebagai berikut.

- a. void delete(int elementNum)
- b. void deleteAll()

Setiap ditambahkan elemen List akan memanjang ke bawah sehingga kemungkinan akan muncul *scroll bar* jika List lebih panjang dari ukuran layar. Ada tiga jenis List yaitu IMPLICIT, EXCLUSIVE dan MULTIPLE. Sebuah List pasti merupakan salah satu dari ketiga jenis tersebut. Implementasi MIDP pada perangkat mobile mempengaruhi tampilan List. Untuk memilih suatu elemen dari List tergantung dari perangkatnya juga. Jika perangkat tersebut memiliki tombol khusus untuk memilih dapat menggunakan itu jika tidak maka terpaksa menggunakan softkey.

List.IMPLICIT memungkinkan pengguna mendapatkan respon begitu dia memilih suatu elemen. Untuk itu sebuah List memerlukan implementasi dari CommandListener. List.EXCLUSIVE memungkinkan user hanya dapat memilih satu elemen dari List. Setiap kali user memilih elemen List

```
listJobs.addCommand(backCmd);
```

---



tidak langsung meresponnya seperti `List.IMPLICIT`. `List.MULTIPLE` memungkinkan user untuk memilih lebih dari satu elemen pada `List`. Tipe ini juga tidak memberikan respon langsung begitu user memilih elemen.

#### **Listing II-8** Contoh penggunaan `List`

```
Command backCmd = new Command("Back", Command.BACK, 1);
List listJobs = new List("Vacancies", Choice.IMPLICIT);
listJobs.append("Programmer", null);
listJobs.append("System Analyst", null);
listJobs.append("GUI Designer", null);
listJobs.append("Network Administrator", null);
listJobs.append("Database Administrator", null);
listJobs.append("Technical Suport", null);
listJobs.setCommandListener(this);
```

**Gambar II-8** Tampilan List pada WTK

## **II.12 TextBox**

TextBox adalah turunan Screen yang memungkinkan user memasukkan atau mengedit teks. TextBox minimal harus memiliki satu Command untuk menentukan cara memproses input dari user.

Ada suatu vendor perangkat mobile yang menyediakan secara otomatis satu Command. Tetapi akan lebih baik jika pengembang perangkat lunak sendiri yang menyediakannya. Cara input user ke dalam perangkat yang beraneka ragam mulai dari input dari tombol sampai menggunakan *connector pen* baik memakai prediktif maupun non-prediktif mode. Selain itu juga dimungkinkan memasukkan karakter-karakter non-latin seperti huruf Yunani dan hiragana.

Format input dan perilaku dari teks yang dimasukkan user sama seperti yang berlaku pada `TextField`. Pada MIDP 2.0 dapat ditentukan untuk menggunakan satu jenis cara input teks dengan memakai *method* sebagai berikut.

```
void setInitialInputMode(String)
```

*Method* `setInitialInputMode (String)` ini dapat mengatur karakter apa yang dapat dimasukkan oleh user. Karakter Unicode diberi tambahan “UCB\_” di depan setiap namanya sesuai dengan yang terdefinisi pada kelas `java.lang.Character.UnicodeBlock`. Input karakter ini berlaku juga pada komponen `TextField`. Meskipun demikian tetap saja pada implementasinya tergantung vendor apakah akan mendukung karakter-karakter tersebut atau tidak. Karakter Unicode yang biasanya diimplementasikan dapat dilihat di tabel dibawah ini.

**Tabel II-3** Karakter unicode yang biasanya diimplementasikan

Karakter Unicode
UCB_BASIC_LATIN
UCB_GREEK
UCB_CYRILLIC
UCB_ARMENIAN
UCB_HEBREW
UCB_ARABIC
UCB_DEVANAGARI
UCB_BENGALI
UCB_THAI
UCB_HIRAGANA
UCB_KATAKANA
UCB_HANGUL_SYLLABLES

**Listing II-9** Contoh pemakaian `TextBox`

```
Command cmdSend = new Command("Send",Command.SCREEN,0);
```

```
TextBox tb = new TextBox("Message","",255,TextField.ANY);  
Display.getDisplay(this).setCurrent(tb);
```



**Gambar II-9 Tampilan TextBox**

```
tb.addCommand(cmdSend);
```

### II.13 Alert

Alert merupakan komponen yang mirip fungsinya seperti *dialog box* pada J2SE. Alert digunakan untuk menampilkan informasi terkait suatu kondisi apakah itu error, konfirmasi maupun kondisi-kondisi lain yang perlu diberitahukan kepada user. Selain pesan dalam Alert juga dimungkinkan terdapat Image, Command ataupun Gauge. Alert dapat dipanggil dari kelas Display dengan menggunakan *method* yang sama dengan memanggil turunan Displayable lainnya yaitu.

```
void setCurrent(Alert alert, Displayable nextDisplayable)
```

Alert mempunyai atribut *timeout* yang menentukan seberapa lama Alert akan ditampilkan pada layar perangkat. *timeout* ini dapat dikendalikan dengan menggunakan *method* sebagai berikut.

```
void setTimeout(int time)
```

Jika *timeout* ini tidak diset maka akan tergantung pada perangkat tempat MIDlet berada berapa nilai *default timeout* ini diatur. Alert dapat diset untuk tampil terus menerus sampai ada interaksi dari user untuk menutupnya dengan menggunakan Alert.FOREVER sebagai argumen pada *method* *setTimeout(int)*. Untuk mengetahui berapa lama perangkat mendefinisikan *timeout* untuk Alert dapat menggunakan *method* sebagai berikut.

```
int getDefaultTimeout()
```

Alert memiliki beberapa jenis tipe untuk digunakan sesuai dengan informasi yang dibawanya. Tipe dari Alert yaitu AlertType antara lain ALARM, CONFIRMATION, ERROR, INFO dan WARNING.

`AlertType.ALARM` biasanya digunakan untuk menginterupsi proses yang sedang terjadi. Contoh penggunaannya pada *push mechanism* misalkan jika ada pesan masuk ataupun reminder dari *Calendar*.

Gambar yang ditampilkan pada Alert dapat didefinisikan sendiri ataupun ditampilkan tergantung dari perangkat yang digunakan. Selain gambar, Alert juga dapat diatur supaya memainkan suara pada saat ditampilkan. Untuk memainkan suara dapat menggunakan *method* `AlertType.playSound()`. Jika hanya ada satu Command pada Alert, maka Command itu otomatis akan dieksekusi jika `timeout` telah terpenuhi. Gauge yang dapat dimasukkan ke dalam Alert adalah Gauge dengan tipe noninteraktif.

#### **Listing II-10 Penggunaan Alert**

```
Alert alert = new Alert("Vacancy", "Sending your application",  
null, AlertType.INFO); alert.setTimeout(4000);  
display.setCurrent(alert);
```



**Gambar II-10** Tampilan Alert pada WTK

### **II.14 Event Handling**

Event handling pada J2ME dibagi menjadi dua jenis yaitu:

- a. Event yang terjadi ketika user mengubah state dari suatu obyek Item pada Form
- b. Event yang terjadi ketika user memilih obyek Command .

Event pertama terjadi misalnya ketika user mengubah teks pada obyek `TextField`, mengubah pilihan pada `ChoiceGroup`, memasukkan tanggal baru pada `DateField` ataupun mengubah nilai dari `Gauge`.

Command digunakan untuk penanganan event atau merespon user. Command juga digunakan jika user memilih Command obyek dari Form, Item maupun sebuah elemen di List.

## II.15 Command

Command adalah obyek yang digunakan sebagai antarmuka respon user perangkat *mobile*. Untuk mendapatkan respon dari user Command sering diasosiasikan dengan *softkey*, tombol atau mungkin area pada *touch screen*. Command pada umumnya digunakan untuk navigasi antar obyek *Displayable* dan untuk melakukan proses atas masukan dari user.

Command merupakan komponen penting dalam penanganan even atau event handling pada J2ME. Penanganan even menggunakan Command sama seperti pada Button/JButton di J2SE yaitu user mendefinisikan obyek Command lalu menambahkan *listener* pada obyek tersebut. Kelas untuk *listener* Command adalah CommandListener.

Command dapat dimasukkan ke dalam obyek *Displayable* menggunakan *method* sebagai berikut  
public void addCommand(Command cmd)

Sedangkan penghapusan dengan menggunakan *method* sebagai berikut

```
public void removeCommand(Command cmd)
```

Dalam satu obyek *Displayable* dapat terdapat lebih dari dua Command. Setiap obyek Command mempunyai tiga atribut yaitu *label*, *type* dan *priority*. *label* pada Command digunakan untuk menampilkan nama Command pada layar perangkat *mobile*.

*type* pada Command digunakan untuk menentukan fungsi dari Command tersebut. *type* ini menentukan *softkey* mana yang akan merepresentasikannya. Implementasi posisi Command (*softkey* yang merepresentasikan) ini tergantung pada vendor perangkatnya. Jenis-jenis *type* dari Command antara lain BACK, CANCEL, EXIT, HELP, OK, SCREEN, ITEM dan STOP. jenis ITEM biasanya diasosiasikan dengan sebuah komponen turunan Item yang terdapat pada Form. Jenis SCREEN diasosiasikan untuk semua komponen yang terdapat pada/dalam turunan obyek Screen.

*priority* menentukan urutan prioritas penampilan suatu Command pada layar. Semakin kecil *priority* maka Command itu akan lebih mudah diakses dibandingkan dengan Command yang nilai *priority*-nya lebih besar.

## II.16 Contoh Aplikasi MIDlet Menggunakan Antarmuka Tingkat Atas

Pada sub-bab ini akan ditunjukkan pembuatan antarmuka pengguna untuk suatu MIDlet yang diberi nama PhotoViewer. Antarmuka yang dibuat harus mengakomodasi tempat untuk menampilkan foto, tanggal dan tempat untuk memberi komentar atas foto yg ditampilkan. Kode program yang ditampilkan pada bab

ini hanyalah untuk menampilkan antarmuka pengguna dan tidak termasuk melakukan pemrosesan masukan pengguna.

**Listing II-11 MIDlet PhotoViewer**

```
import java.io.IOException; import
java.util.Date; import
javax.microedition.lcdui.*; import


---


javax.microedition.midlet;

public class PhotoViewer extends MIDlet implements CommandListener{
    private Form form;
    private Command nextCmd, exitCmd, saveCmd;
    private Display display;    private DateField
dateField;    private TextField txtField;
    private int photoIndex;    private ImageItem
imageItem;

    public PhotoViewer() {    display =
        Display.getDisplay(this);    form =
        new Form("Photo Viewer");    nextCmd =
        new Command("Next", Command.SCREEN,1);
        exitCmd = new Command("Exit",
        Command.EXIT, 3);    saveCmd = new
        Command("Save", Command.SCREEN,1);
        txtField = new TextField("Caption
:", "", 32, TextField.ANY);    dateField
= new DateField("Date",
        DateField.DATE);
        dateField.setDate(new
        Date());photoIndex = 0;    imageItem =
        new ImageItem(null, getImage(),
        ImageItem.LAYOUT_CENTER, "Photo
Viewer");
        form.append(imageItem);form.append(txt
        Field);    form.append(dateField);
        form.addCommand(nextCmd);
        form.addCommand(saveCmd);
        form.addCommand(exitCmd);
        form.setCommandListener(this);
        display.setCurrent(form);
    }
    private Image getImage() {
        photoIndex++;
        String photoname = "/photo"+photoIndex+".png";
        try{
            return Image.createImage(photoname);
        } catch(IOException ioe){ioe.printStackTrace();}
        return null;    }
    private void nextImage(){imageItem.setImage(getImage());}
    public void commandAction(Command cmd, Displayable d){
        if(cmd==exitCmd)notifyDestroyed();
        else if(cmd==nextCmd){nextImage();}
        else{/*Save Image attributes */}
```

```
    }
    public void startApp() {}
    public void pauseApp() {}
    public void destroyApp(boolean d) {}
}
```

---



**Gambar II-11** Tampilan MIDlet PhotoViewer

MIDlet PhotoViewer hanya terdiri dari satu kelas yaitu PhotoViewer. Kelas PhotoViewer ini sekaligus turunan dari kelas `java.microedition.midlet.MIDlet`. Pada MIDlet ini digunakan beberapa komponen antarmuka pengguna tingkat atas seperti `Command`, `Form`, `TextField`, `DateField` dan `ImageItem`. `Form` berfungsi menampung komponen-komponen lainnya. `TextField` digunakan untuk menyediakan tempat bagi user untuk memasukkan komentar atas foto yang ditampilkan. `DateField` digunakan untuk menampilkan tanggal foto. Tampilan pada gambar XXX diatas adalah tampilan pada WTK. Kemungkinan jika MIDlet PhotoViewer ini dijalankan pada emulator perangkat *mobile* dari vendor (Nokia, BenQ-Siemens atau Sony Ericsson) tampilannya akan berbeda.

Pada kelas PhotoViewer ini bagian tiga *method* utama MIDlet (`startApp()`, `pauseApp()` dan `destroyApp()`) tidak mempunyai implementasi. Hal ini dikarenakan semua kode yang diperlukan sudah ditulis dan dipanggil pada konstruktur. Sehingga implementasinya menggunakan implementasi turunan dari kelas orang tuanya yaitu kelas `javax.microedition.midlet.MIDlet`.

## III Antarmuka Tingkat Bawah (Canvas)

---

### III.1 Sekilas Canvas

Untuk membuat sebuah tampilan dengan kreasi sendiri, pengembang membutuhkan pengetahuan dasar tentang Canvas. Canvas menyediakan media untuk menciptakan tampilan yang sesuai dengan keperluan aplikasi atau keinginan pengguna. Media ini mendukung kapabilitas di dalam membuat beragam bentuk, teks, dan gambar (statis atau animasi), yang pada dasarnya merupakan kemampuan obyek Graphics.

Canvas memiliki kemampuan berinteraksi dengan pengguna melalui mekanisme *event handling*. *Event handling* ini dapat berupa antisipasi terhadap *key event*, *game action* dan *pointer event*. Perangkat komunikasi, telepon seluler ataupun PDA, memiliki resolusi tampilan yang sangat beragam.

Untuk mengetahui dimensi Canvas terkait dengan tampilan perangkat mobile, disediakan *method* `getWidth()` dan `getHeight()`. MIDP 2.0 menyediakan fungsionalitas *full screen mode*. Tetapi tidak semua perangkat mobile mendukung hal ini. Setting mode ini dapat dilakukan dengan memanggil `setFullScreenMode(boolean)`.

### III.2 Paint dan Repaint

Untuk menampilkan konten Canvas, pengembang aplikasi perlu memanggil *method* `paint()`. Implementasi MIDP menyertakan obyek Graphics sebagai argumen `paint()`. Graphics memiliki *method-method* untuk menggambar bentuk 2D, teks, dan gambar pada Canvas. Untuk menggunakan Canvas, sebuah kelas yang mewarisi kelas abstrak Canvas harus dibuat. Tipikal format kelas yang mengimplementasi Canvas sebagai berikut.

---

**Listing III-1** Format implementasi Canvas

```
import javax.microedition.lcdui.*;

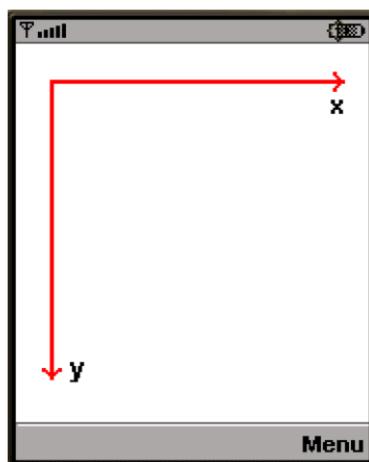
public class MyCanvas extends Canvas {
    public void paint(Graphics g) {
        // Kreasi tampilan Canvas menggunakan obyek g
    }
}
```

---

*Method* `paint()` tidak dapat dipanggil secara langsung karena tidak memiliki obyek `Graphics` yang sesuai sebagai argumen `paint()`. Sebagai gantinya, *method* `repaint()` digunakan untuk memberi tahu implementasi MIDP bahwa sudah waktunya untuk menggambar konten pada `Canvas`.

### III.3 Ruang Koordinat

Penggambaran pada `Canvas` dilakukan dalam ruang koordinat yang berbasis pada piksel perangkat *mobile*. Secara default, titik awal koordinat berada pada sudut kiri atas `Canvas`. Nilai pada koordinat x meningkat ke arah kanan, sedangkan nilai pada koordinat y meningkat ke arah bawah.



**Gambar III-1** Representasi koordinat

Titik awal koordinat dapat disesuaikan dengan memanggil `translate()` dari kelas `Graphics`. Untuk mengetahui lokasi titik koordinat hasil translasi terhadap titik koordinat default, digunakan `getTranslateX()` dan `getTranslateY()`.

### III.4 Bentuk

`Graphics` memiliki *method-method* untuk menggambar dan mengisi bentuk-bentuk yang sederhana.

**Tabel III-1** *Method* untuk menggambar bentuk-bentuk sederhana

Shape Outline	Filled Shape
<code>drawLine(int x1, int y1, int x2, int y2)</code>	-
-	<code>fillTriangle(int x1, int y1, int x2, int y2, int x3, int y3)</code>
<code>drawRect(int x, int y, int width, int height)</code>	<code>fillRect(int x, int y, int width, int height)</code>

drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)
drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)	fillArc(int x, int y, int width, int height, int startAngle, int arcAngle)

Berikut akan ditunjukkan kode untuk menggambar berbagai bentuk dengan menggunakan obyek Graphics.

**Listing III-2** Kelas ShapeCanvas.java

---

```
import javax.microedition.lcdui.*; public class
ShapeCanvas extends Canvas {     public void
paint(Graphics g) {         int w =
getWidth();         int h = getHeight();
g.setColor(0xffffffff);
g.fillRect(0,0,w,h);
g.setColor(0x000000);
int z = 10;
g.drawLine(z,z,z+10,z);
int z += 10;
g.drawRect(z,z,20,20);
int z += 25;
g.fillRoundRect(z,z,20,20,5,5);
int z += 20;
g.drawArc(z,z,20,20,0,360);
int z += 20;
g.fillArc(z,z,20,20,0,270);
int z += 20;
g.fillTriangle(z,z,z+20,z,z+40);
}
}
```

---

**Listing III-3** Kelas ShapeMidlet.java

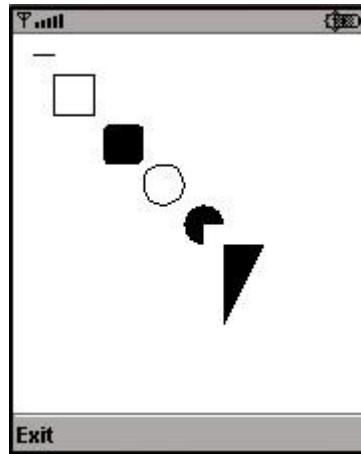
---

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class ShapeMidlet extends MIDlet {    public
void startApp() {
ShapeCanvas s = new ShapeCanvas();
s.addCommand(new Command("Exit", Command.EXIT, 0));
}}
```

}

---



**Gambar III-2** Hasil eksekusi kode ShapeCanvas.java

```
s.setCommandListener(new CommandListener() {  
    public void commandAction(Command c, Displayable d) {  
        notifyDestroyed();  
    }  
});  
Display.getDisplay(this).setCurrent(s);  
}  
public void destroyApp(boolean unconditional) {}  
  
public void pauseApp() {}
```

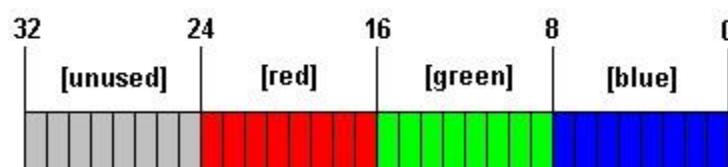
### III.5 Warna

Graphics menggunakan warna yang ditentukan untuk menggambar *outline* bentuk, mengisi bentuk, dan menggambar teks. Warna direpresentasikan sebagai kombinasi merah (*red*), hijau (*green*), dan biru (*blue*), dengan nilai 8 bit untuk tiap komponen warna. Warna dapat diset menggunakan *method* berikut

```
public void setColor(int RGB)
```

*Method* tersebut mengkombinasikan semua nilai komponen warna dalam sebuah nilai integer/bilangan bulat. *Method* alternatif untuk mengeset warna dapat menggunakan nilai tiap komponen warna sebagai nilai integer dengan nilai dari 0 hingga 255. *Method* tersebut yaitu:

```
public void setColor(int red, int green, int blue)
```



**Gambar III-3** Alokasi warna pada bilangan bulat (*integer*)

### III.6 Tipe Garis

Graphics menggunakan tipe garis (*stroke style*) yang ditentukan untuk menggambar berbagai *outline* bentuk dan garis. Tipe garis dari kelas Graphics adalah :

- a. SOLID sebagai nilai default
- b. DOTTED

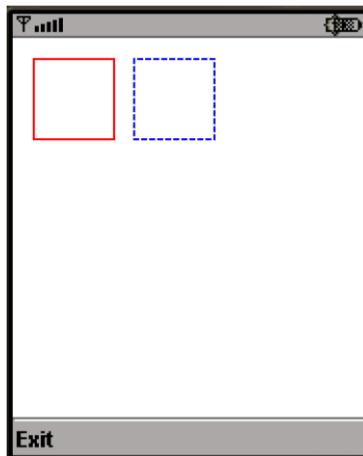
Untuk mengganti tipe garis, dapat digunakan `setStrokeStyle()`. Kode berikut digunakan untuk menggambar sebuah persegi empat dengan garis yang solid dan sebuah persegi empat dengan garis putus-putus.

---

**Listing III-4** Method `paint()` yang menggambar dua persegi

```
public void paint(Graphics g) {  
    g.setColor(0xffffffff);  
    g.fillRect(0, 0, getWidth(), getHeight());  
    g.setColor(0xff0000);  
    g.drawRect(10, 10, 40, 40);  
    g.setStrokeStyle(Graphics.DOTTED);  
    g.setColor(0x0000ff);  
    g.drawRect(60, 10, 40, 40);  
}
```

---



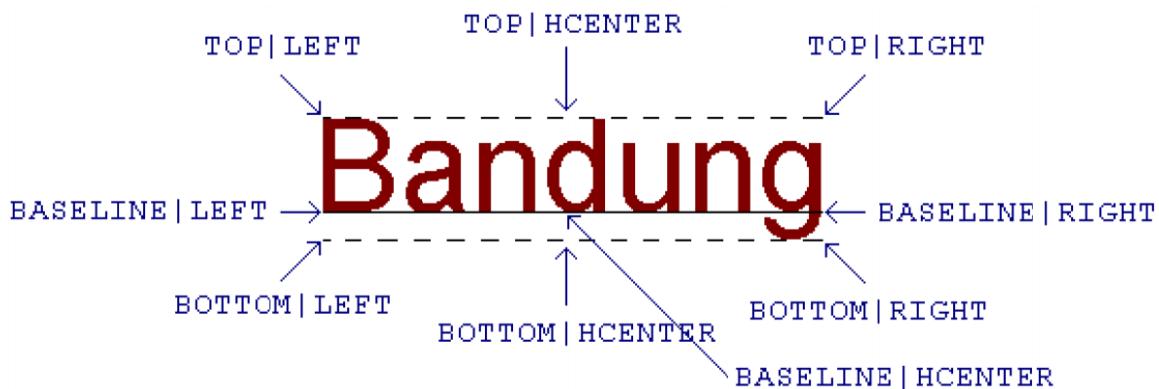
**Gambar III-4** Tampilan persegi yang dibuat

### III.7 Teks

Penggambaran teks dilakukan berdasarkan *anchor point*. *Anchor point* menentukan secara tepat di mana teks akan digambar. *Anchor point* dideskripsikan dengan komponen horisontal dan vertikal, yang didefinisikan sebagai kontanta di kelas Graphics.

Graphics memiliki 4 *method* yang berbeda untuk menggambar teks. Keempat *method* itu yaitu:

- a. public void drawChar(char character, int x, int y, int anchor)
- b. public void drawChars(char[] data, int offset, int length, int x, int y, int anchor)
- c. public void drawString(String str, int x, int y, int anchor)
- d. public void drawSubstring(String str, int offset, int length, int x, int y, int anchor)



Gambar III-5 Posisi *anchor point* untuk teks

Berikut ini akan ditampilkan contoh kode yang menampilkan teks di layar. Teks yang dituliskan akan menggunakan *anchor* dan menampilkan tiga teks di tiga tempat di layar. Kode ini akan masuk dalam kelas yang diberi nama TextCanvas.

---

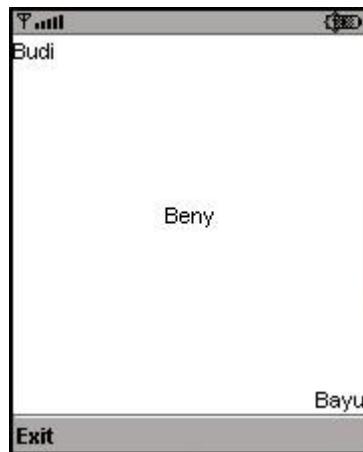
**Listing III-5** File TextCanvas.java

```
import javax.microedition.lcdui.*;  
  
public class TextCanvas extends Canvas {  
    public void paint(Graphics g) {        int  
        w = getWidth();        int h = getHeight();  
        g.setColor(0xffffffff);  
        g.fillRect(0,0,w,h);  
        g.setColor(0x000000);  
        g.drawString("Budi", 0, 0,  
        Graphics.TOP|Graphics.LEFT);  
        g.drawString("Beny", w/2, h/2,  
        Graphics.BASELINE|Graphics.HCENTER);  
    }  
}
```

```
    g.drawString("Bayu", w, h, Graphics.BOTTOM|Graphics.RIGHT);  
}  


---


```



**Gambar III-6** Tampilan kode TextCanvas.java

Untuk mengetahui dimensi teks, *method-method* berikut dapat digunakan.

- a. public int getHeight()
- b. public int charWidth(char ch)
- c. public int charsWidth(char ch, int offset, int length)
- d. public int stringWidth(String str)
- e. public int substringWidth(String str, int offset, int length)

Berikut akan disajikan sebuah MIDlet yang menampilkan sebuah tulisan yang dilingkupi oleh kotak. Kotak yang melingkupi tulisan tersebut ditentukan dimensinya dengan pengukuran teks. Kelas MIDlet diberi nama TextSizeCanvas.

---

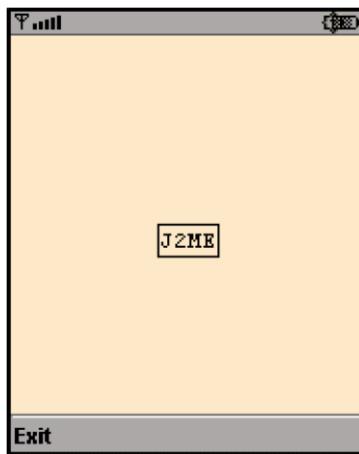
**Listing III-6** File TextSizeCanvas.java

```
import javax.microedition.lcdui.*;  
  
public class TextSizeCanvas extends Canvas {  
    private Font font;    public  
    TextSizeCanvas() {  
        font = Font.getFont(Font.FACE_MONOSPACE, Font.STYLE_PLAIN,  
                           Font.SIZE_MEDIUM);  
    }  
    public void paint(Graphics g) {  
        int w = getWidth();    int h =  
        getHeight();  
        g.setColor(0xffeecc);  
        g.fillRect(0, 0, w, h);  
        g.setColor(0x000000);
```

```
    g.setFont(font);
String s = "J2ME";
int sWidth = font.stringWidth(s);
int sHeight = font.getHeight();
int x = (w - sWidth) / 2;      int y =
h / 2;
g.drawString(s, x, y, Graphics.TOP|Graphics.LEFT);

}
}
```

---



**Gambar III-7** Tampilan TextSizeCanvas.java  
g.drawRect(x-1, y, sWidth+2, sHeight);

### III.8 Font

Font direpresentasikan oleh nama font (*font face*), tipe font (*font style*), dan ukuran font (*font size*). Masing-masing representasi Font tersebut berbentuk konstanta berupa atribut *static* pada kelas Font. Konstanta dari representasi Font dapat dilihat pada table dibawah. Untuk nama Font pada J2ME tidak dapat ditambahkan seperti pada lingkungan desktop. *Method* `setFont()` digunakan untuk memberitahu obyek dari kelas `Graphics` menggunakan Font baru bagi teks-teks selanjutnya yang akan digambar.

**Tabel III-2** Representasi konstanta font

Representasi	Konstanta
Nama Font ( <i>font face</i> )	FACE_SYSTEM FACE_MONOSPACE FACE_PROPORIONAL
Tipe font ( <i>font face</i> )	STYLE_PLAIN STYLE_BOLD STYLE_ITALIC STYLE_UNDERLINED
Ukuran font ( <i>font size</i> )	SIZE_SMALL SIZE_MEDIUM SIZE_LARGE



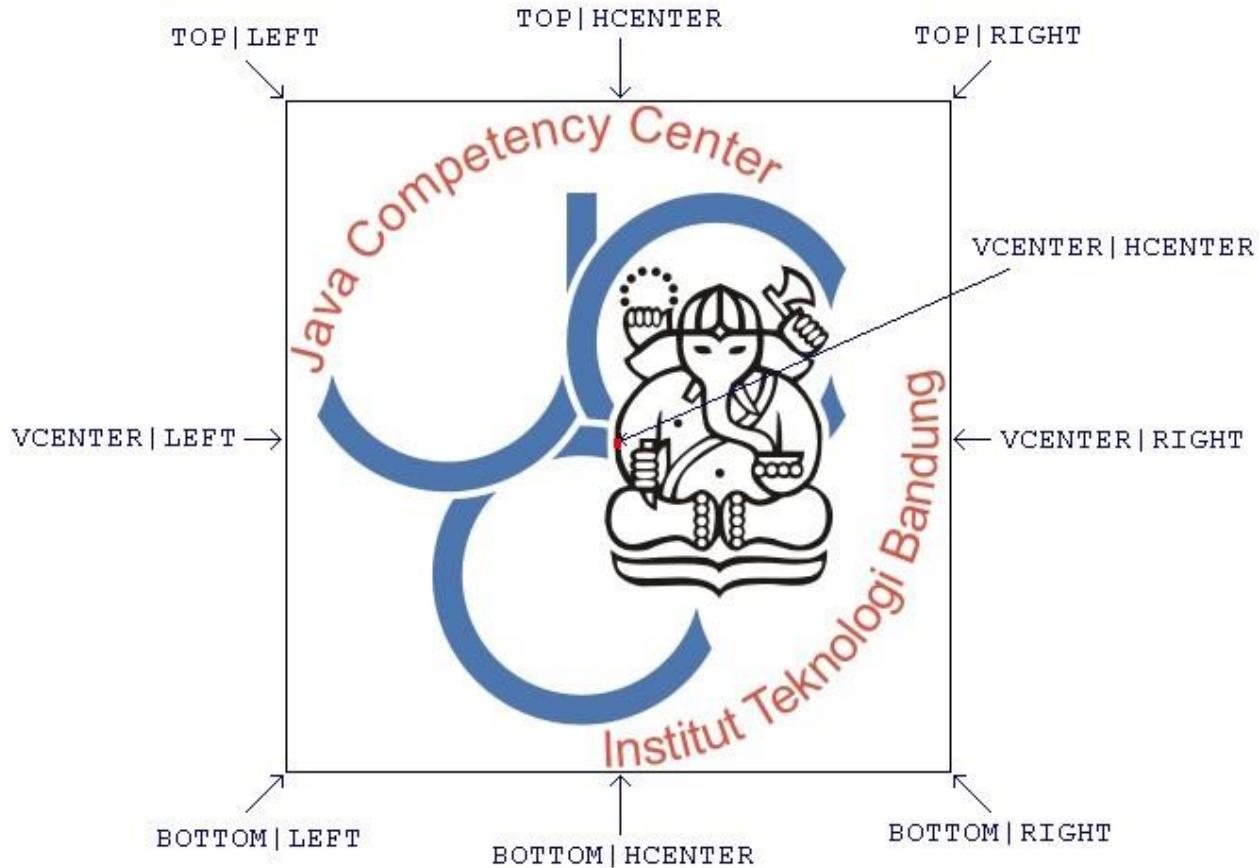
**Gambar III-8** Font dengan variasi konstanta

### III.9 Gambar

Graphics menggunakan *method* berikut untuk menggambar sebuah gambar.

```
public void drawImage(Image img, int x, int y, int anchor)
```

Sama halnya dengan menggambar teks, *anchor point* digunakan untuk menentukan secara tepat di mana sebuah gambar akan digambar pada Canvas.



Gambar III-9 Posisi *anchor point* untuk gambar

### III.10 Transformasi

Gambar Graphics memiliki *method* `drawRegion()` untuk menggambar area sebuah gambar dan dapat juga melakukan manipulasi gambar pada waktu yang bersamaan.

```
public void drawRegion(Image src, int x_src, int y_src, int width,  
int height, int transform, int x_dest, int y_dest, int anchor)
```

Salah satu manipulasi gambar adalah transformasi. Ada delapan jenis transformasi yang dapat dilakukan oleh MIDP. Masing-masing transformasi itu dapat dilihat pada table XXX. Parameter transform dapat berupa berbagai bentuk transformasi yang didefinisikan pada kelas Sprite (bagian dari Game API yang akan dibahas pada bab 7).

**Tabel III-3** Jenis-jenis transformasi

Jenis Transformasi	Keterangan
TRANS_NONE	Tidak melakukan transformasi
TRANS_ROT90	Melakukan rotasi (berputar) 90 derajat searah jarum jam (clockwise)
TRANS_ROT180	Melakukan rotasi (berputar) 180 derajat searah jarum jam (clockwise)
TRANS_ROT270	Melakukan rotasi (berputar) 270 derajat searah jarum jam (clockwise)
TRANS_MIRROR	Melakukan pencerminan terhadap sumbu vertical
TRANS_MIRROR_ROT90	Melakukan pencerminan terhadap sumbu vertical lalu rotasi (berputar) 90 derajat searah jarum jam (clockwise)
TRANS_MIRROR_ROT180	Melakukan pencerminan terhadap sumbu vertical lalu rotasi (berputar) 180 derajat searah jarum jam (clockwise)
TRANS_MIRROR_ROT270	Melakukan pencerminan terhadap sumbu vertical lalu rotasi (berputar) 270 derajat searah jarum jam (clockwise)

Berikut akan ditunjukkan contoh MIDlet yang melakukan transformasi terhadap suatu gambar. Pada MIDlet dibawah ditunjukkan suatu gambar awal dan gambar yang telah di rotasi dengan sudut sebesar 90 derajat.

---

**Listing III-7** File TransformationCanvas.java

```

import java.io.IOException; import
javax.microedition.lcdui.*; import
javax.microedition.lcdui.game.Sprite;

public class TransformationCanvas extends Canvas {
    public void paint(Graphics g) {
Image img = null;
    try {img = Image.createImage("/ponsel.gif");}
    catch(IOException e) { return; }      int
imgWidth = img.getWidth();      int

```

```

        imgHeight = img.getHeight();      int w =
        getWidth(); int h = getHeight();
        g.setColor(0xffffffff); g.fillRect(0,0,w,h);
        g.setColor(0x000000); w = w / 2;
        g.drawString("Gambar Awal",w,0,Graphics.TOP|Graphics.HCENTER);
        g.drawImage(img,w,20,Graphics.TOP|Graphics.HCENTER);
        g.drawString("Transformasi: TRANS_ROT90",           w,
30+imgHeight, Graphics.TOP|Graphics.HCENTER);
        g.drawRegion(img, 0, 0, imgWidth, imgHeight, Sprite.TRANS_ROT90,
                     w, 50+imgHeight, Graphics.TOP|Graphics.HCENTER);
    }
}

```

---



**Gambar III-10** Tampilan hasil TransformationCanvas.java

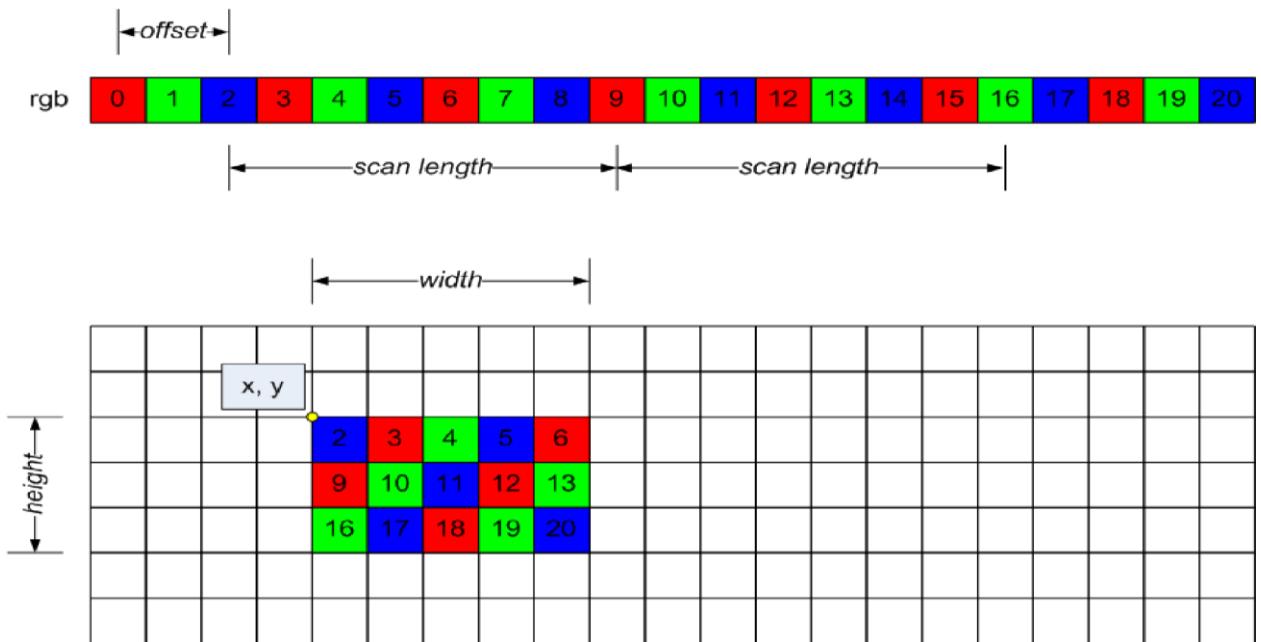
### III.11 Gambar Sebagai Array Integer

Gambar dapat direpresentasikan dalam sebuah array integer, di mana setiap elemen array mendeskripsikan warna untuk setiap piksel gambar. *Method* berikut dapat digunakan untuk membentuk gambar dari sebuah *array integer*.

```

public void drawRGB(int[] rgbData, int offset, int scanLength, int
x, int y, int width, int height, boolean processAlpha)

```



**Gambar III-11** Penjelasan visual dari parameter *method* drawRGB ()

Parameter `offset` menspesifikasikan indeks nilai warna untuk piksel pertama area gambar target. Parameter `scanLength` menentukan jumlah elemen array RGB antara piksel pertama untuk tiap baris area gambar target. Contoh dapat dilihat pada listing beikut.

---

**Listing III-8** Representasi gambar dalam *integer*

```
int[] rgb = {
    0xff0000, 0x00ff00, 0x0000ff, 0xff0000, 0x00ff00, 0x0000ff,
    0xff0000, 0x00ff00, 0x0000ff, 0xff0000, 0x00ff00, 0x0000ff,
    0xff0000, 0x00ff00, 0x0000ff, 0xff0000, 0x00ff00, 0x0000ff,
    0xff0000, 0x00ff00, 0x0000ff};
g.drawRGB(rgb, 2, 7, 4, 2, 5, 3, false);
```

---

Contoh lengkap suatu MIDlet yang menggambar pada layar dengan representasi array integer dapat dilihat pada kode program MIDlet dibawah ini.

---

**Listing III-9** File ArrayCanvas.java

```
import java.io.IOException; import
javax.microedition.lcdui.*;

public class ArrayCanvas extends Canvas {
    private int[] rgb = {
        0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff,
```

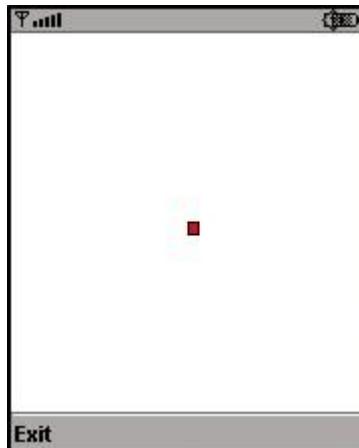
```

0x000000, 0x000000, 0x000000, 0x000000, 0x000000, 0x000000,
0x000000, 0xff0000, 0xff0000, 0xff0000, 0xff0000, 0x000000,
0x000000, 0x000000, 0x000000, 0x000000, 0x000000, 0x000000,
0xfffffff, 0xfffffff, 0xfffffff, 0xfffffff, 0xfffffff};

    public void paint(Graphics g) {
    int w = getWidth();      int h =
    getHeight();
        g.setColor(0xffffffff);
        g.fillRect(0,0,w,h);
        g.setColor(0x000000);
        g.drawRGB(rgb, 6, 6, w/2, h/2, 6, 7, false);
    }
}

```

---



**Gambar III-12** Representasi gambar dalam bentuk *array of integer*

### III.12 Clipping

*Clipping* pada dasarnya membatasi penggambaran. Semua penggambaran di luar bentuk atau area *clipping* tidak akan ditampilkan. Untuk mendapatkan informasi area *clipping*, *method* berikut dapat digunakan.

- a. public int getClipX()
- b. public int getClipY()
- c. public int getClipWidth()
- d. public int getClipHeight()

Untuk memodifikasi area clipping, *method* berikut dapat digunakan

- a. public void setClip(int x, int y, int width, int height)
- b. public void clipRect(int x, int y, int width, int height)

*Method* `setClip()` secara langsung mengupdate area *clipping* yang baru, sedangkan *method* `clipRect()` mengupdate area *clipping* yang baru berdasarkan hasil perpaduan (irisian) antara area *clipping* eksisting dengan area *clipping* yang ditentukan pada argumen *method*. Selanjutnya diberikan contoh penggunaan *clipping* untuk membatasi area Canvas yang dapat digunakan.

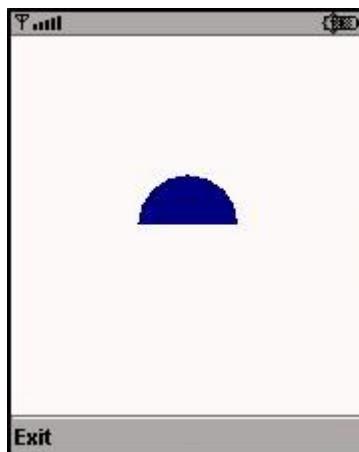
---

**Listing III-10** File ClippingCanvas.java

```
import javax.microedition.lcdui.*;

public class ClippingCanvas extends Canvas {
    private Image image;    public ClippingCanvas
    () {      try {
        image = Image.createImage(50, 50);
        Graphics g = image.getGraphics();
        g.setColor(0, 0, 128);
        g.fillArc(0, 0, 50, 50, 0, 360);
    } catch (Exception e) {} }
    protected void paint(Graphics g) {
if (image != null) {
        g.setColor(255, 255, 255);
        int w = getWidth();      int h
= getHeight();
        g.fillRect(0, 0, w, h);
        g.setClip(0, 0, w, h/2);
        g.drawImage(image, w/2, h/2, Graphics.VCENTER|Graphics.HCENTER);
    }
}
}
```

---



**Gambar III-13** Tampilan ClippingCanvas.java

### III.13 Key Event

Canvas menyertakan sekumpulan *method* untuk mengantisipasi interaksi dengan setiap tombol di suatu perangkat *mobile*. *Method* berikut selalu dipanggil ketika pengguna menekan atau melepaskan sebuah tombol.

- a. protected void keyPressed(int keyCode)
- b. protected void keyReleased(int keyCode)

Kode tombol (*key code*) yang diberikan kepada kedua *method* di atas merupakan salah satu dari konstanta-konstanta pada kelas Graphics, dimulai dari KEY\_NUM0 hingga KEY\_NUM9, KEY\_STAR, dan KEY\_POUND. Untuk mendapatkan deskripsi tekstual dari *key code* yang diberikan, *method* getKeyname() dapat digunakan.

Game API (GameCanvas) memiliki mekanisme yang berbeda di dalam mengantisipasi *key event*. Canvas menerapkan mekanisme *callback*, sedangkan GameCanvas menerapkan mekanisme polling di mana pengembang aplikasi dapat langsung memeriksa status tombol pada perangkat mobile dengan menggunakan *method* yang disediakan. Penjelasan lebih lanjut dapat dilihat pada bab 7.



**Gambar III-14** Ragam bentuk *keyboard* perangkat *mobile*

MIDlet berikut menggambarkan deskripsi tekstual tombol yang ditekan pada bagian tengah obyek Canvas.

---

#### **Listing III-11** File KeyEventCanvas.java

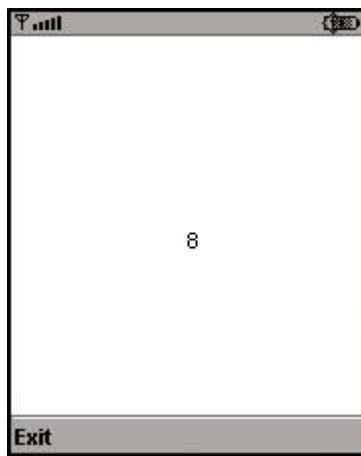
```
import javax.microedition.lcdui.*;  
  
public class KeyEventCanvas extends Canvas {  
    private String keyName = "";    public void
```

```

paint(Graphics g) {      int w = getWidth();
int h = getHeight();
g.setColor(0xffffffff);
g.fillRect(0,0,w,h);
g.setColor(0x000000);
g.drawString(keyName, w/2, h/2, Graphics.TOP|Graphics.LEFT);
}
protected void keyPressed(int keyCode) {
keyName = getKeyName(keyCode);
repaint();
}
}

```

---



**Gambar III-15** Canvas yang merespon masukan dari pengguna (KeyEventCanvas.java)

### III.14 GameAction

*Key code* cukup tergantung dari perangkat mobile. Oleh karena itu, MIDP menawarkan sebuah abstraksi sederhana yang dinamakan *game action*, yang memudahkan pemetaan *key event* dari pengguna dengan event yang berguna bagi game atau aplikasi lain dengan antarmuka yang khusus. *Game action* merupakan solusi untuk memetakan tombol-tombol pada perangkat mobile dengan sekumpulan tombol video game, seperti yang ditemukan pada platform game (Nintendo, Sega, PlayStation, dan lain-lain). Konsep : memberikan sebuah masukan kepada *method* `getGameAction()` untuk mendapatkan sebuah *game action*, yang dapat berupa kontanta berikut:

**Tabel III-4** Game key

UP	LEFT	GAME_C
RIGHT	GAME_A	GAME_D
DOWN	GAME_B	FIRE

Sebagai contoh, pada telepon seluler, kita dapat memetakan UP dengan tombol 2, DOWN dengan 8, LEFT dengan 4, dan RIGHT dengan 6.

*Game action* membantu pengembang aplikasi dari keperluan memetakan sendiri. Implementasi MIDP menyediakan pemetaan yang sangat beralasan untuk perangkat *mobile* yang mendukung Java. Aplikasi berikut melakukan konversi *key code* ke *game action* dan menampilkannya pada Canvas.

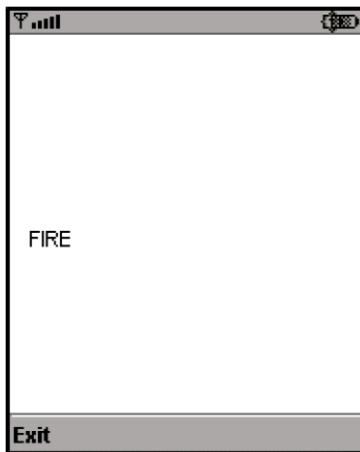
---

**Listing III-12** File GameActionCanvas.java

```
import javax.microedition.lcdui.*;  
  
public class GameActionCanvas extends Canvas {  
    private String gameAction = "[Tekan Tombol]";  
    public void paint(Graphics g) {  
        int w = getWidth();  
        int h = getHeight();  
        g.setColor(0xffffffff);  
        g.fillRect(0, 0, w, h);  
        g.setColor(0x000000);  
        g.drawString(gameAction, 10, h/2, Graphics.TOP|Graphics.LEFT);  
    }  
    protected void keyPressed(int keyCode) {  
        switch(getGameAction(keyCode)) {  
            case UP:  
                gameAction = "UP"; break;  
            case DOWN:  
                gameAction = "DOWN"; break;  
            case LEFT:  
                gameAction = "LEFT"; break;  
            case RIGHT:  
                gameAction = "RIGHT"; break;  
            case FIRE:  
                gameAction = "FIRE"; break;  
            case GAME_A:  
                gameAction = "GAME_A"; break;  
            case GAME_B:  
                gameAction = "GAME_B"; break;  
            case GAME_C:  
                gameAction = "GAME_C"; break;  
            case GAME_D:  
                gameAction = "GAME_D"; break;  
            default:  
                gameAction = ""; break;  
        }  
        repaint();  
    }  
}
```

```
    }  
}
```

---



**Gambar III-16** Tampilan Canvas yang merespon *Game key FIRE* (*GameActionCanvas.java*)

### III.15 Pointer Event

*Pointer event* merupakan event yang dihasilkan oleh perangkat *pointing*, seperti *stylus*, yang umumnya dijumpai pada perangkat kategori PDA atau smart phone.



**Gambar III-17** Perangkat *mobile* yang menggunakan *pointer*

Untuk mengetahui apakah perangkat *mobile* yang dimiliki mendukung *pointer event*, *method hasPointerEvents()* dan *hasPointerMotionEvents()* dari kelas *Canvas* dapat digunakan.

Jika *pointer event* dimungkinkan, maka *method* berikut akan dipanggil ketika *pointer* ditekan atau dilepaskan.

- a. `protected void pointerPressed(int x, int y)`
- b. `protected void pointerReleased(int x, int y)`

Jika *pointer motion event* dimungkinkan, maka *method* berikut akan dipanggil ketika pengguna menggerakkan stylus pada layar perangkat *mobile*.

```
protected void pointerDragged(int x, int y)
```

### III.16 Double Buffering

*Double Buffering* merupakan teknik yang umum digunakan untuk mengurangi efek *flicker* (efek transisi atau penggambaran yang membutuhkan waktu lama) pada proses penggambaran atau animasi. Untuk mengetahui apakah sebuah obyek *Canvas* menggunakan teknik *double buffering* atau tidak, *method* `isDoubleBuffered()` dapat digunakan. Jika implementasi tidak menggunakan *double buffering*, pengembang aplikasi dapat melakukannya sendiri dengan proses sebagai berikut. Membuat *off-screen image* dengan menggunakan `Image.createImage(int width, int height)`. Mendapatkan obyek *Graphics* yang melakukan penggambaran pada *off-screen image* dengan memanggil *method* `getGraphics()` dari obyek *Image* melakukan penggambaran pada *off-screen image* menggunakan obyek *Graphics*. Pada *method* `paint()` dari kelas *Canvas*, *method* `drawImage()` digunakan untuk menempatkan *off-screen image* pada obyek *Canvas*.

Contoh kode kelas *ClippingCanvas* (pada pembahasan bagian *Clipping*) mengimplementasikan proses *double buffering*.

### III.17 Animasi

Animasi merupakan bentuk dinamisasi tampilan terhadap waktu. Tipikal implementasi animasi dilakukan oleh sebuah *thread* yang berbeda dari *thread* tampilan (*system-owned user-interface thread*). Sebagai catatan bahwa implementasi antarmuka MIDP memiliki *thread* sendiri yang menangani *method* antarmuka dan proses penggambaran pada layar tampilan. Berikut diberikan contoh pembuatan animasi sederhana pada *Canvas*, yaitu menampilkan bentuk persegi empat dimulai dari tidak kelihatan secara bertahap menuju ukuran persegi empat yang diinginkan.

**Listing III-13** File *AnimationCanvas.java*

---

```
import javax.microedition.lcdui.*;

public class AnimationCanvas extends Canvas implements Runnable {
    private int w, h, stepX, stepY, index;

    public AnimationCanvas() {      w =
getWidth(); h = getHeight();
stepX = w/20;   stepY = h/20;
}
```

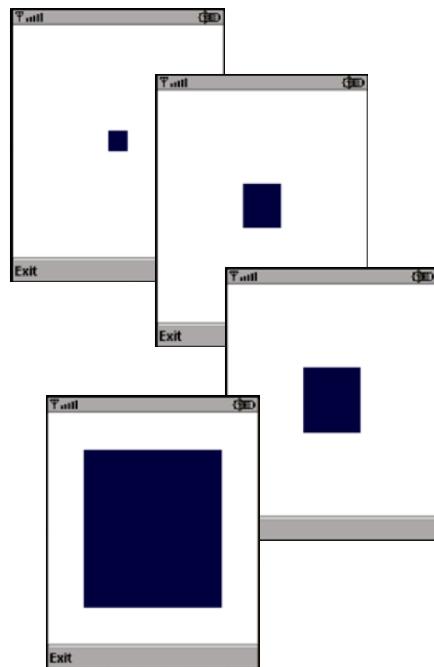
```

protected void showNotify() {new Thread(this).start();}

public void run() {
try {
    while(index<=10) {
Thread.sleep(90);      index++;
repaint();
}
} catch(InterruptedException e) {}
}
protected void paint(Graphics g) {
g.setColor(255, 255, 255);
g.fillRect(0, 0, w, h);
g.setColor(0, 0, 64);
g.fillRect(30, 30, w-60, h-60);
g.setColor(255, 255, 255);
int incX = index * stepX;      int
incY = index * stepY;
g.fillRect(0, 0, w/2 - incX, h);
g.fillRect(w/2 + incX, 0, w/2 - incX, h);
g.fillRect(0, 0, w, h/2 - incY);
g.fillRect(0, h/2 + incY, w, h/2 - incY);
}
}

```

---



**Gambar III-18** Tampilan animasi



## IV Basis Data pada J2ME

---

### IV.1 Sekilas Tentang Basis Data pada J2ME

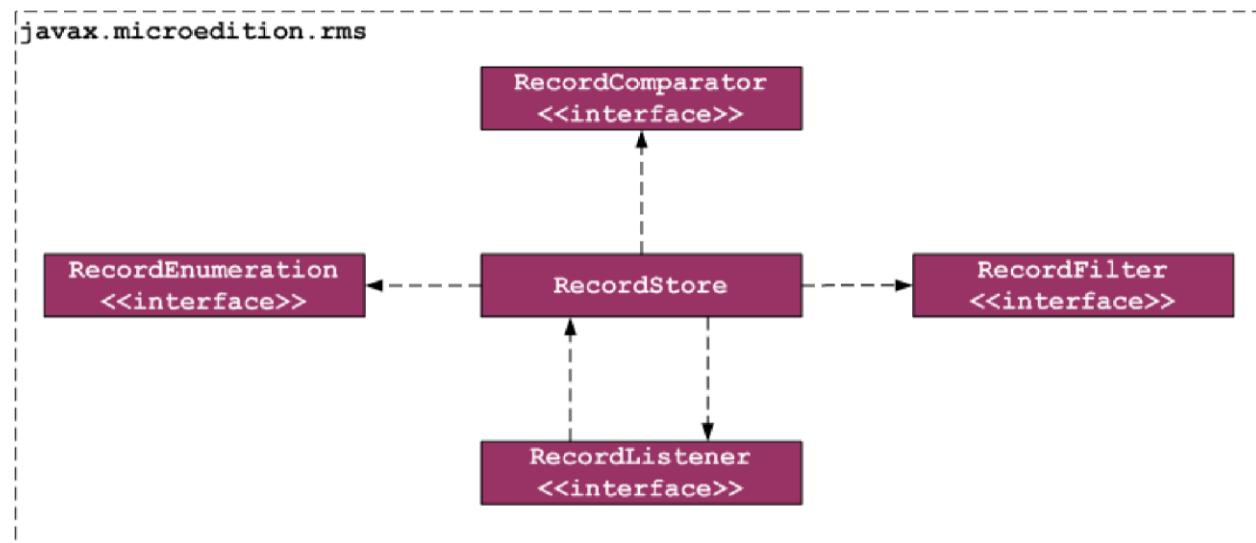
Penyimpanan informasi/data pada perangkat mobile di J2ME dapat menggunakan dua cara yaitu:

- Menyimpan data pada suatu file yang diletakkan pada direktori `res`
- Menggunakan *Record Management System* (RMS)

Cara pertama mempunyai kelemahan yaitu data yang telah ditulis pada file tersebut tidak dapat diubah lagi ketika MIDlet sedang *runtime*. Cara kedua, yaitu menggunakan RMS, adalah cara yang lazim digunakan untuk penyimpanan data pada perangkat *mobile*. RMS memungkinkan proses penyimpanan dan pengambilan data dapat dilakukan seperti pada basis data konvensional yaitu *Create, Retrieve, Update* dan *Delete* atau yang biasa disingkat CRUD. Pada bab ini akan dibahas mengenai RMS saja.

Spesifikasi RMS telah ada sejak MIDP 1.0. Hal ini menjanjikan bahwa perangkat *mobile* yang MIDP compliant memiliki RMS. RMS menjamin integritas data yang disimpannya. Akan tetapi perlu diperhatikan *thread* yang menggunakan data pada RMS tersebut.

Data yang disimpan oleh RMS secara fisik dapat ditempatkan di lebih dari satu tempat. Tempat-tempat yang biasanya dipakai antara lain pada Ram perangkat *mobile* dan media penyimpanan yang dapat digunakan oleh perangkat *mobile* seperti flash disk. Data yang disimpan dalam RMS disebut *record*. *Record* merupakan data yang berbentuk *array of byte*. Pustaka RMS terdapat dalam *package* `javax.microedition.rms`.

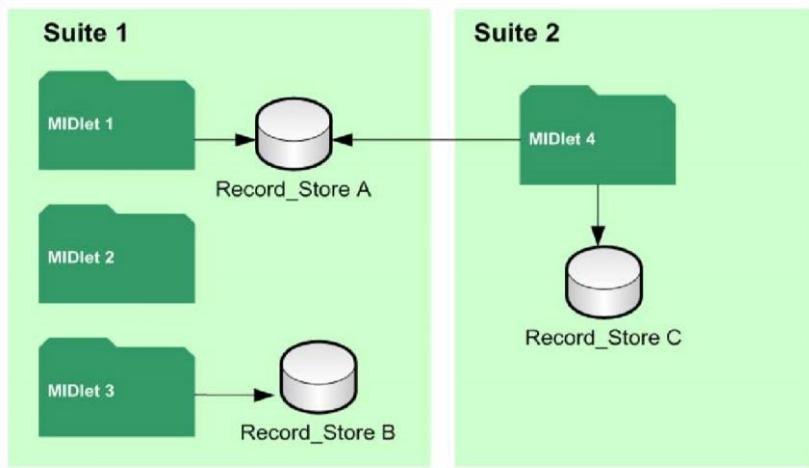


Gambar IV-1 Package Record Management System

## IV.2 RecordStore

Kelas RecordStore pada RMS berfungsi menyediakan fitur untuk menyimpan, mengupdate, mengambil dan menghapus data pada perangkat *mobile*. Kelas RecordStore merepresentasikan sebuah lokasi penyimpanan data permanen. Satu RecordStore dapat digunakan oleh lebih dari satu MIDlet baik pada saat bersamaan maupun terpisah. Berdasarkan spesifikasi MIDP, vendor perangkat *mobile* bertanggung jawab pada pemeliharaan integritas obyek RecordStore.

Antar MIDlet dapat berbagi RecordStore. Pada versi MIDP lama, sebelum MIDP 2.0, MIDlet hanya dapat mengakses obyek RecordStore selama MIDlet tersebut adalah pemilik obyek RecordStore atau masih dalam satu MIDlet *suite* dengan MIDlet pemilik obyek RecordStore. Pada MIDP 2.0 terdapat fitur baru di mana RecordStore dapat dibagi dengan MIDlet lain dari MIDlet *suite* yang berbeda dengan MIDlet pemilik RecordStore. Gambar IV-1 mengilustrasikan dua *MIDlet suite* berbagi RecordStore.



**Gambar IV-2** Hubungan antara MIDlet dengan suatu RecordStore

Dalam kelas RecordStore didefinisikan tiga *static method* untuk membuat dan membuka obyek RecordStore. *Method-method* tersebut adalah sebagai berikut:

- a. static RecordStore openRecordStore(String recordStoreName, boolean createIfNecessary)
- b. static RecordStore openRecordStore(String recordStoreName, boolean createIfNecessary, int authMode, boolean writable)
- c. static RecordStore openRecordStore(String recordStoreName, String vendorName, String suiteName)

Masing-masing *method* diatas memiliki kegunaan sendiri-sendiri. *Method* pertama merupakan *method* untuk membuka dan membuat RecordStore yang dibuat oleh MIDlet itu sendiri atau MIDlet lain yang masih dalam satu MIDlet *suite*. *Method* kedua merupakan *method* untuk membuat dan mengakses RecordStore yang *shared* antar MIDlet *suite*. *Method* ketiga khusus digunakan untuk mengakses

RecordStore yang *shared* antar MIDlet *suite*. Pemanggilan *method* openRecordStore() dapat menimbulkan *exception* dikarenakan beberapa hal seperti kapasitas memori perangkat *mobile* yang sudah penuh (RecordStoreFullException) ataupun *internal error* (RecordStoreException).

Mekanisme dari *method* pertama ketika dipanggil adalah mencari RecordStore dengan nama sesuai dengan parameter recordStoreName yang dimasukkan. Jika RecordStore dengan nama tersebut ditemukan maka RecordStore itu akan dibuka. Jika tidak terdapat RecordStore dengan nama tersebut maka akan dilihat nilai dari parameter createIfNecessary yang diberikan. Jika nilainya true maka akan dibuat sebuah RecordStore baru dengan nama seperti pada parameter recordStoreName. Jika nilainya false maka tidak akan dibuat RecordStore baru. Pemberian nama untuk RecordStore harus memenuhi persyaratan sebagai berikut:

- a. Panjang sebuah nama 32 *Unicode character*
- b. Nama harus memperhatikan huruf kapital dan huruf non kapital (*case sensitive*)
- c. Nama harus unik di dalam sebuah *MIDlet suite*

*Method* kedua digunakan untuk membuka (mungkin juga menciptakan) RecordStore yang dapat diakses oleh MIDlet baik dalam MIDlet *suite* yang sama ataupun MIDlet *suite* yang berbeda. RecordStore yang tercipta dari pemanggilan *method* ini menjadi milik MIDlet *suite* yang menciptakannya meskipun dapat diakses oleh MIDlet *suite* lain. Pada *method* kedua didefinisikan mode akses RecordStore pada parameter authMode. Mode akses ada dua jenis yaitu akses yang hanya mengijinkan pemilik RecordStore dan mode akses yang mengijinkan MIDlet *suite* lain untuk menggunakan RecordStore. Parameter writeable ditujukan untuk MIDlet *suite* lain yang mengakses RecordStore. Jika RecordStore hanya dapat diakses MIDlet *suite* pemiliknya maka nilai parameter ini diabaikan. Pemilik RecordStore selalu dapat menulis pada RecordStore, jadi parameter writeable hanya akan berpengaruh pada MIDlet *suite* lain apakah diijinkan menulis pada RecordStore atau tidak. Berikut akan disajikan potongan kode program yang menunjukkan cara pemanggilan *method* kedua.

**Listing IV-1** Contoh pembuatan RecordStore yang dapat diakses MIDlet suite lain

---

```
import javax.microedition.rms.*;
... try{
    RecordStore rs =
        RecordStore.openRecordStore("MyData",true,
        RecordStore.AUTHMODE_ANY,false);
    rs.deleteRecord(1);      rs.closeRecordStore();
} catch{RecordStoreException rse} {
    //tangani error di sini }
```

---

**Tabel IV-1** Konstanta Akses Informasi RecordStore

Nilai Parameter	Deskripsi
AUTHMODE_PRIVATE	Akses hanya dibatasi pada suite tempat RecordStore bersangkutan ditempatkan
AUTHMODE_ANY	Mengijinkan akses oleh semua MIDlet di suite manapun. Gunakan parameter ini untuk membuat Obyek RecordStore yang sharable

*Method* ketiga digunakan untuk membuka RecordStore milik MIDlet *suite* lain. Suatu MIDlet *suite* dapat mengakses RecordStore milik MIDlet *suite* yang lain hanya jika RecordStore tersebut diijinkan untuk diakses oleh MIDlet *suite* lain selain pemiliknya. Jika tidak mendapatkan ijin maka SecurityException akan terjadi. Parameter vendorName dan suiteName harus disesuaikan dengan Midlet-Vendor dan Midlet-Name pada file JAD (lebih lanjut tentang file JAD dijelaskan di Bab 8). Jika *method* ini digunakan oleh MIDlet dalam MIDlet *suite* pemilik RecordStore maka efeknya akan sama dengan menggunakan *method* pertama tanpa membuat RecordStore baru openRecordStore(recordStoreName, false). Berikut disajikan potongan kode program contoh pemanggilan *method* ketiga

**Listing IV-2** Contoh pengambilan RecordStore yang Dibuat MIDlet suite lain

---

```
import javax.microedition.rms.*;
... try{
    RecordStore rs =
    RecordStore.openRecordStore("MyData",
        "Sun Microsystems", "Jupiter");
    byte[] data = rs.getRecord(1);
    rs.closeRecordStore(); }catch(RecordStoreException
rse) {
    //tangani error di sini }
```

---

Setelah obyek RecordStore yang telah dibuka dapat diperoleh informasi berkenaan dengan obyek tersebut. Informasi tersebut antara lain mengenai nama, ukuran, versi, dan waktu terakhir obyek RecordStore dimodifikasi. *Method-method* untuk mengetahui informasi RecordStore dapat dilihat pada table IV-2.

**Tabel IV-2** Daftar *method* untuk mengambil informasi RecordStore

Nama Method	Deskripsi
String getName()	Mengembalikan nama dari obyek RecordStore
int getSize()	Mengembalikan ukuran ruang dalam byte yang ditempati oleh obyek RecordStore. Ukuran tersebut meliputi <i>overhead</i> termasuk <i>overhead</i> yang diperlukan untuk implementasi struktur data yang diterapkan oleh obyek RecordStore.
int getSizeAvailable()	Mengembalikan tempat yang masih tersisa pada RecordStore
long getLastModified()	Mengembalikan informasi waktu kapan RecordStore dimodifikasi terakhir kali. Penulisan informasi waktu ini menggunakan format yang sama dengan yang digunakan oleh <code>System.currentTimeMillis()</code> .
int getVersion()	Mengembalikan versi RecordStore. Setiap obyek RecordStore dimodifikasi, seperti waktu ketika <i>record</i> ditambah, dimodifikasi, atau dihapus, maka versinya ditambah satu.
int getNumRecords()	Mengembalikan jumlah data/ <i>record</i> yang saat ini ada dalam sebuah obyek RecordStore.

Obyek RecordStore yang tidak digunakan lagi sebaiknya segera dihapus untuk menyediakan tempat bagi RecordStore yang lainnya. Untuk penghapusan RecordStore terdapat *static method* dari kelas RecordStore sebagai berikut

```
static void deleteRecordStore(String recordStoreName)
```

Sebelum menghapus suatu obyek RecordStore pastikan tidak ada MIDlet yang sedang/masih menggunakan atau memerlukan data dalam RecordStore tersebut. RecordStoreException akan dihasilkan jika RecordStore yang akan dihapus sedang dibuka oleh MIDlet lain baik dalam satu MIDlet *suite* atau lain MIDlet *suite*. RecordStoreNotFoundException akan muncul ketika RecordStore yang akan dihapus tidak tersedia.

Untuk mengetahui RecordStore apa saja yang dimiliki oleh suatu MIDlet *suite* maka RecordStore menyediakan sebuah *static method* untuk mendapatkan nama RecordStore dari suatu MIDlet *suite*.

*Method* tersebut adalah sebagai berikut.

```
static String[] listRecordStores()
```

RecordStore yang ditampilkan hanyalah RecordStore milik dari MIDlet suite tempat MIDlet pemanggil *static method* tersebut.

### IV.3 Penyimpanan Data

Setiap data yang tersimpan dalam RecordStore disebut *record*. *Record* tersimpan dalam format *array of byte* (byte []). Ada beberapa hal mengenai karakteristik RMS yang harus diperhatikan yaitu:

- a. RMS menyimpan *record* dalam format *byte array* (byte []). Informasi yang disimpan dalam obyek RecordStore harus dalam bentuk *byte array* (byte []). Hal ini menjamin kesesuaian dengan *platform* lain (*cross platform compability*). Jadi, keterbatasan tidak hanya pada informasi apa yang akan disimpan pada obyek RecordStore, tapi juga pada bagaimana informasi tersebut direpresentasikan. Untuk itu, terlepas dari informasi yang akan disimpan, informasi tersebut harus dikonversi terlebih dahulu data ke dalam format byte [].
- b. RMS mengisi nilai ID yang unik. Pada saat informasi disimpan di dalam obyek RecordStore, RMS mengisi ID unik untuk setiap *record*. Hal ini merupakan cara untuk mengidentifikasi setiap *record* baru yang telah ditambahkan. RMS mengisi nilai ID ini dalam urutan mulai dari satu (1) dan ID tersebut tidak dapat digunakan kembali (*reuse*). Hal ini berarti setelah *record* dihapus dari obyek RecordStore, ID yang telah diisi untuk *record* tersebut tidak lagi *valid* untuk digunakan. ID suatu *record* diset secara otomatis dan tidak dapat diganti.

Untuk memasukkan *record* ke dalam RecordStore, dalam RecordStore terdapat sebuah *method* sebagai berikut.

```
int addRecord(byte[] data, int offset, int numByte)
```

*Method* diatas mengembalikan sebuah bilangan bulat yang merupakan ID dari *record* yang dimasukkan. ID ini nantinya bermanfaat untuk pengambilan *record* dari RecordStore. Data yang dimasukkan direpresentasikan dalam *array of byte* dalam method diatas adalah parameter *data*. *offset* menandakan indeks byte awal dari *array* yang akan dimasukkan dan *numByte* merupakan jumlah byte yang dimasukkan ke RMS mulai dari byte indeks ke-*offset*.

Berikut akan diberikan potongan kode program yang melakukan proses pengisian *record* ke dalam RecordStore.

**Listing IV-3** Contoh pengisian *record* ke dalam RecordStore

---

```
import javax.microedition.rms.*;
...
try
{
    RecordStore rs =
        RecordStore.openRecordStore("MyData, true);
```

```

String data = "Welcome to my Class";
byte[] dataArray = data.getBytes();      int
recID =
    rs.addRecord(dataArray,0,dataArray.length);
    rs.closeRecordStore();
}catch{RecordStoreNotOpenException rsnoe){
    // tangani error di sini
}catch(RecordStoreException rse){
    //tangani error di sini
}

```

---

...

#### IV.4 Pengambilan Data

RecordStore memiliki dua *method* untuk mengambil suatu *record* dengan menggunakan indeks (ID)nya. Dua *method* tersebut yaitu:

- a. public byte[] getRecord(int recordID)
- b. public int getRecord(int recordID, byte[] buffer, int offset)

Perbedaan antara kedua *method* tersebut selain tipe kembalian dan parameternya yaitu jika *method* pertama memberikan salinan/copy dari obyek yang tersimpan pada RecordStore sedangkan *method* kedua memberikan obyek yang tersimpan dalam RecordStore.

**Listing IV-4** Contoh pengambilan *record* pada RecordStore

---

```

import javax.microedition.rms.*;
... try{
    RecordStore rs = RecordStore.openRecordStore("MYData", true);
    byte[] data = rs.getRecord(1);    rs.closeRecordStore();
}catch{RecordStoreNotOpenException rsnoe}{// tangani error di sini
}catch(RecordStoreException rse){//tangani error di sini }
...

```

---

Dalam RecordStore didefinisikan beberapa *method* untuk mengetahui informasi suatu *record*.

*Method-method* tersebut antara lain:

- a. int getRecordSize(int ID)  
Untuk mengetahui ukuran *record* yang ID-nya disebutkan sebagai parameter.
- b. int getNextRecordID()  
Untuk mengetahui nilai recordID berikutnya yang akan ditambahkan ke obyek RecordStore. *Method* ini perlu digunakan untuk mendapatkan informasi tentang *record* akhir dari sebuah obyek RecordStore.

## IV.5 Mengambil Record Secara Sekuensial

Pada sub-bab sebelumnya telah dibahas mengenai pengambilan *record* dalam *RecordStore* dengan menggunakan ID dari *record*. ID dari *record* memang diset secara sekuensial akan tetapi pengambilan *record* secara sekuensial dengan menggunakan ID-nya secara sekuensial didalam suatu *loop* memungkinkan terjadinya *RecordNotFoundException*. Hal ini dikarenakan mungkin saja ada ID yang tidak dipergunakan lagi oleh suatu *record* karena *record* yg bersangkutan telah dihapus.

*RecordStore* menyediakan sebuah *interface* untuk melakukan iterasi pengambilan *record* yang ada dalam suatu *RecordStore*. Interface itu adalah *RecordEnumeration*. *Interface RecordEnumeration* menyediakan satu set API yang mirip dengan *interface Enumeration* pada J2SE dalam paket *java.util*. *Interface RecordEnumeration* memfasilitasi proses pengulangan dalam obyek *RecordStore* tanpa harus mengetahui nilai record ID. Namun, dalam penggunaan *RecordEnumeration* perlu berhati-hati karena *interface* ini dapat menyebabkan *overhead/beban kerja* yang cukup besar pada perangkat *mobile*, khususnya dalam hal kinerja/*processing* dan penggunaan memori.

*ID Record* merupakan *primary key* untuk sebuah *record*. *Interface RecordEnumeration* menjaga urutan *record ID* di dalam obyek *RecordStore*. *Enumerator* dapat melakukan iterasi semua *record* bersamaan dengan penggunaan *interface* tambahan seperti *RecordComparator* dan *RecordFilter*. Jika *RecordFilter* disertakan, *enumerator* dapat melakukan iterasi sampai ke subset *record*.

Kelas *RecordStore* menyediakan sebuah *method* yang mengembalikan interface

*RecordEnumeration*. Berikut *method* lengkap dari *enumerateRecords()*.

```
RecordEnumeration enumerateRecords(RecordFilter rf,  
                                  RecordComparator rc, boolean b)
```

*Record* diatas membutuhkan tiga parameter yaitu *RecordFilter*, *RecordComparator* dan sebuah nilai *boolean*. Parameter-parameter ini pada umumnya untuk memasukkan beberapa fungsional *query* seperti yang biasa dijumpai pada basis data konvensional. Penjelasan mengenai fungsi masing-masing parameter dapat dilihat pada Tabel IV-3.

**Tabel IV-3** Daftar *method* untuk mengambil informasi *RecordStore*

Parameter	Deskripsi
RecordFilter rf	Jika parameter ini tidak null, parameter ini menentukan subset <i>record</i> <i>RecordStore</i> yang akan digunakan. Dalam hal ini record yang sesuai filter yang akan dienumerasi.

RecordComparator rc	Jika parameter ini tidak null, parameter ini digunakan untuk menentukan kembalian <i>record</i> apakah berurut atau tidak sesuai dengan komparatornya.
boolean b	Jika parameter ini bernilai true, maka <i>enumerator</i> akan tetap menyimpan enumerasi terakhir dengan segala perubahan di <i>recordnya</i> . Jika parameter ini bernilai false, enumerasi terakhir tidak disimpan. Enumerator akan mengembalikan suatu <i>integer</i> primitif yang menunjukkan <i>record ID</i> dari <i>record</i> yang terhapus.

*Interface RecordEnumeration menggunakan method nextRecord() dan previousRecord()* untuk membaca *record* berikutnya (*next*) dan sebelumnya (*previous*) di dalam suatu enumerasi. Kedua *method* tersebut adalah sebagai berikut.

- a. byte[] nextRecord()
- b. byte[] previousRecord()

*Method* tersebut mengembalikan salinan dari *record* berikutnya (*next record*) dan sebelumnya (*previous record*), dimana *next* dan *previous* didefinisikan oleh *interface RecordComparator* dan *RecordFilter* yang dibentuk pada saat obyek enumerasi dibuat. Selain itu ada dua *method* lagi untuk mengetahui ID dari *record* selanjutnya (*next*) dan Id dari *record* sebelumnya (*previous*) yaitu sebagai berikut.

- a. int nextRecordId()
- b. int previousRecordId()

*Byte array* yang dikembalikan melalui proses enumerasi adalah salinan *record*. Perubahan pada *array* ini tidak mengubah *record* di obyek *RecordStore*. Setelah memanggil *method* nextRecord() atau previousRecord(), enumerasi selanjutnya berpindah ke *record* berikut yang mungkin masih tersedia. Penggalan kode program berikut menunjukkan contoh penggunaan *interface RecordEnumeration*.

#### **Listing IV-5** Contoh pengambilan *record* menggunakan RecordEnumeration

---

```
import javax.microedition.rms.*;
...
RecordEnumeration re = rs.enumerateRecords(null,null,false);

while(re.hasNextElement()){
byte[] data = re.nextRecord();
// Lakukan sesuatu dengan data
}
```

---

## IV.6 Updating Suatu Data

Memperbarui atau *updating* suatu *record* pada RecordStore terdapat suatu *method* yang dapat digunakan yaitu sebagai berikut.

```
void setRecord(int recordId, byte[] data, int offset,
int numBytes)
```

Parameter *recordId* adalah ID dari *record* yang akan di update sedangkan parameter *data* adalah *data* baru yang akan disimpan dalam *record*. Parameter *offset* adalah indeks awal *data* untuk mulai disalin ke *record*. Parameter Jumlah *byte* yang akan disalin ke *record* dihitung dari *offset*.

Mekanisme *updating* data konvensional yang terdiri dari penghapusan data lalu memasukkan data baru dapat juga dilakukan pada RMS. Hanya saja perlu diketahui bahwa data baru yang masuk memiliki ID yang berbeda dengan data lama yang dihapus.

Berikut ini penggalan kode program untuk memperbarui *record* yang ada dengan menggunakan metod *setRecord()*.

**Listing IV-6** Updating *record*

---

```
import javax.microedition.rms.*;
...
try {
    RecordStore rs = RecordStore.openRecordStore("MyData", true);
    String data = "Welcome to J2ME";
    byte[] dataArray =
data.getBytes();
    rs.setRecord(1,dataArray,0,dataArray.length);
    rs.closeRecordStore();
} catch(RecordStoreNotOpenException rsnoe) {
    // tangani error di sini
} catch(RecordStoreException rse) {
    //tangani error di sini
}
...
```

---

## IV.7 Penghapusan Data

Untuk menghapus suatu *record* dalam RecordStore, RecordStore menyediakan *method* untuk menghapusnya yaitu sebagai berikut.

```
void deleteRecord(int recordID)
```

Proses penghapusan suatu *record* akan membuat ID dari *record* tersebut tidak dapat digunakan. Oleh karena itu perlu diperhatikan agar setelah proses penghapusan ID dari *record* tersebut tidak lagi dipanggil

terutama pada saat proses pengambilan suatu *record* karena akan menyebabkan *RecordNotFoundException*.

**Listing IV-7** Contoh penghapusan *record* pada RecordStore

---

```
import javax.microedition.rms.*;  
...  
try{  
    RecordStore rs =  
        RecordStore.openRecordStore("MyData", true);  
    rs.deleteRecord(1);  
    rs.closeRecordStore(); } catch (RecordStoreException  
rse) {  
    //Tangani error di sini  
}
```

---

## IV.8 Contoh MIDlet Menggunakan RMS

Pada sub-bab ini akan ditampilkan sebuah MIDlet yang menampilkan kode-kode pengoperasian RMS pada perangkat *mobile*. Pada contoh program berikut akan ditampilkan sebuah kelas *UserOptionRS.java* yang menjadi penghubung operasi RMS untuk suatu MIDlet. Selanjutnya akan ditampilkan potongan kode dari MIDlet yang menggunakan *OptionRS.java* bernama *UserOptionMIDlet.java*.

**Listing IV-8** File *UserOptionRS.java*

---

```
import javax.microedition.rms.RecordStore; import  
javax.microedition.rms.RecordStoreException;  
  
public class UserOptionRS {  
  
    private String storeName = "Option";  
    private RecordStore rsUserOption;  
    private byte[] currentUserOption;  
    public UserOptionRS() {  
        try {  
            rsUserOption = RecordStore.openRecordStore(storeName, true);  
        } catch (RecordStoreException e) {  
            e.printStackTrace();  
        }  
    }  
  
    public byte[] readOption() {  
        try {  
            if(currentUserOption == null)  
                currentUserOption = rsUserOption.getRecord(1);  
        } catch (RecordStoreException e) {  
    }
```

---

```

        e.printStackTrace();
    }
    return currentUserOption;
}

public void writeOption(byte[] option) {
    try {
if(recordExists()) rsUserOption.setRecord(1, option, 0, option.length);
else rsUserOption.addRecord(option, 0, option.length);
    } catch(RecordStoreException e) {
e.printStackTrace();
    }
    currentUserOption = option;
}

public boolean recordExists() {
    try {
        if (rsUserOption.getNumRecords() == 1) return true;
    } catch(RecordStoreException e) {
e.printStackTrace();
    }
    return false;
}

public void close() {
    try {
        rsUserOption.closeRecordStore();
    } catch(RecordStoreException e) {
        e.printStackTrace();
    }
}
}

```

---

#### **Listing IV-9 MIDlet UserOptionMidlet.java**

```

import javax.microedition.lcdui.Form; import
javax.microedition.lcdui.TextField; import
javax.microedition.lcdui.Command; import
javax.microedition.lcdui.CommandListener; import
javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable; import
javax.microedition.midlet;

import java.io.IOException; import
java.io.ByteArrayInputStream; import
java.io.ByteArrayOutputStream; import
java.io.DataInputStream; import
java.io.DataOutputStream;

public class UserOptionMidlet extends MIDlet implements CommandListener {

```

```

private Form      form;
private TextField tfName, tfAlias;
private Command   cmdRetrieve, cmdSave, cmdExit;

private UserOptionRS userStore;

public UserOptionMidlet() {
    userStore = new UserOptionRS();

    tfName  = new TextField("Name", "", 15, TextField.ANY);
    tfAlias = new TextField("Alias", "", 15, TextField.ANY);

    cmdSave     = new Command("Save",     Command.SCREEN, 1);
    cmdRetrieve = new Command("Retrieve", Command.SCREEN, 2);      cmdExit
    = new Command("Exit",      Command.EXIT,   1);

    Form form = new Form("User Option");
    form.append(tfName);      form.append(tfAlias);

    form.addCommand(cmdSave);
    form.addCommand(cmdRetrieve);      form.addCommand(cmdExit);

    form.setCommandListener(this);
    Display.getDisplay(this).setCurrent(form);
}

protected void startApp() {}

protected void pauseApp() {}

protected void destroyApp(boolean force) {}

public void commandAction(Command c, Displayable d) {
if (c == cmdSave) {      saveOption();
tfName.setString("");      tfAlias.setString("");      }
else if (c == cmdRetrieve) {      retrieveOption();
} else if (c == cmdExit) {      notifyDestroyed();
}
}

private void retrieveOption() {
try{
    ByteArrayInputStream bOptions = new
        ByteArrayInputStream(userStore.readOption());
    DataInputStream dOptions = new DataInputStream(bOptions);
    tfName.setString(dOptions.readUTF());
    tfAlias.setString(dOptions.readUTF());
    dOptions.close();      bOptions.close();
} catch(IOException e) { e.printStackTrace(); }
}

private void saveOption() {
}

```

```
try {
    ByteArrayOutputStream bOptions = new ByteArrayOutputStream();
    DataOutputStream dOptions = new DataOutputStream(bOptions);
    dOptions.writeUTF(tfName.getString());
    dOptions.writeUTF(tfAlias.getString());
    userStore.writeOption(bOptions.toByteArray());
    dOptions.close();          bOptions.close();
} catch(IOException e) { e.printStackTrace(); }
}
}
```

---



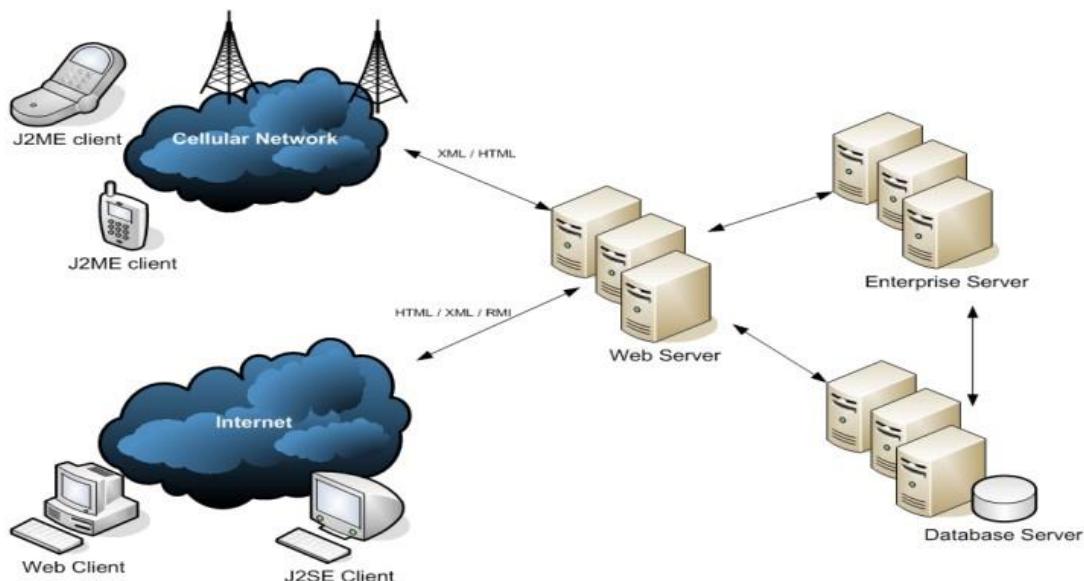
**Gambar IV-3** Hasil eksekusi UserOptionMIDlet.java

# V Pengaksesan Jaringan

## V.1 Lingkungan Perangkat *Mobile*

Salah satu kelebihan dari perangkat *mobile* adalah kemampuannya untuk membentuk koneksi ke suatu jaringan. Kemampuan untuk membentuk koneksi ditambah dengan kemampuan untuk mengirim dan menerima data memberikan peluang bagi pengembang untuk mengembangkan suatu aplikasi bergerak yang menarik.

Perancang *J2ME platform* mempertimbangkan berbagai protokol dan teknologi yang saat ini sedang berkembang. Teknologi J2ME mengembangkan sebuah lingkup pengembangan sehingga teknologi ini dapat berintegrasi dengan aplikasi berbasis web yang saat ini ada atau aplikasi lain yang mendukung protokol HTTP. Gambar V-1 menunjukkan lingkungan MIDP yang berintegrasi dengan jaringan saat ini ada untuk mempertukarkan informasi.



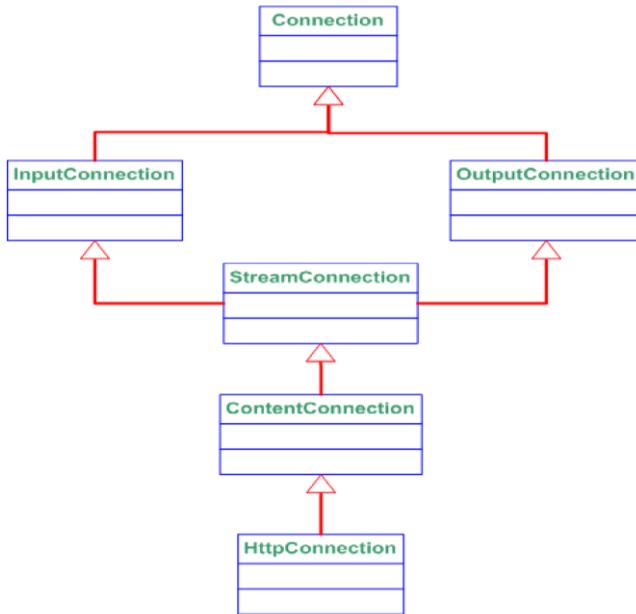
Gambar V-1 Arsitektur jaringan perangkat *mobile*

## V.2 Generic Connection Framework (GCF)

MIDP menyediakan sekumpulan kelas dan *interface* bernama Generic Connection Framework (GCF) untuk membentuk koneksi. Pada MIDP 1.0 koneksi yang dapat digunakan hanyalah HTTP. Pada MIDP 2.0 koneksi HTTP diperluas dengan menambah suatu fitur koneksi HTTP yang aman bernama HTTPS (HTTP Secure). HTTPS ini memungkinkan pertukaran data melalui jalur yang aman. Selain HTTPS ada beberapa koneksi yang didefinisikan pada MIDP 2.0.

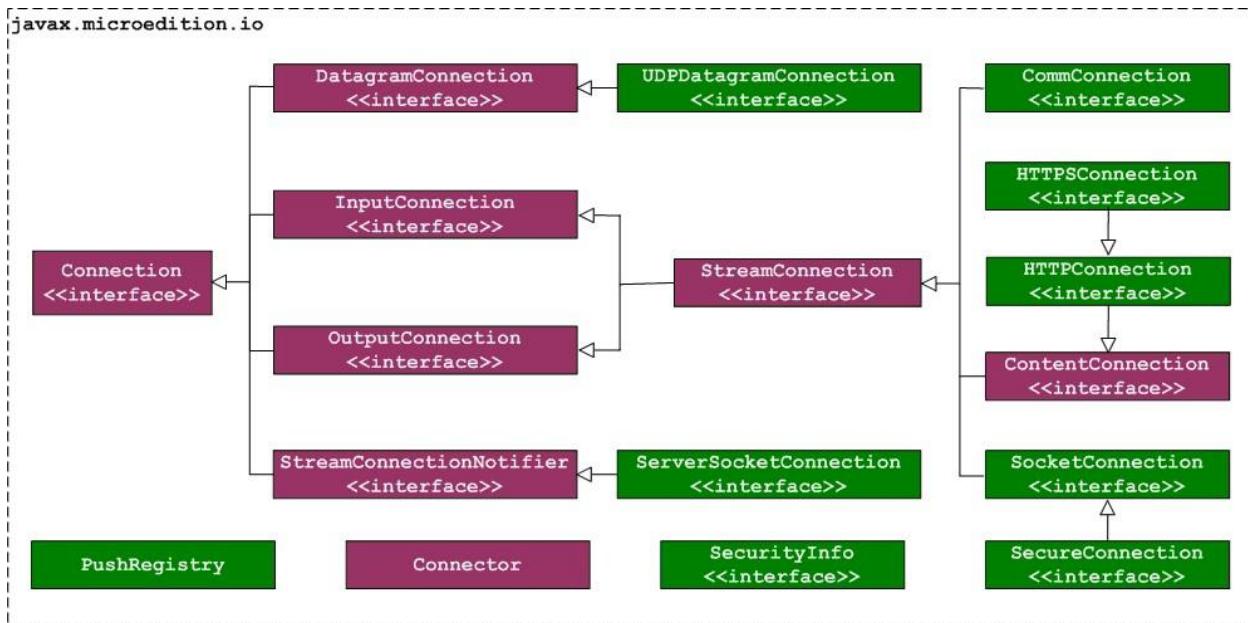
Paket J2SE seperti paket `java.net` dan `java.io` menawarkan kelas-kelas untuk jaringan yang sangat lengkap. Namun, API ini terlalu besar untuk diterapkan di lingkungan J2ME. Untuk itu, CLDC menawarkan sebuah *subset* kelas J2SE `java.io` yang dapat diterapkan di lingkungan perangkat bergerak.

Di MIDP paket `javax.microedition.io` melengkapi paket `java.io` CLDC dan mengimplementasikan GCF yang didefinisikan oleh CLDC. Paket tersebut terdiri dari kelas `Connector`, kelas `ConnectionNotFoundException`, dan hierarki *interface* berikut ini.



Gambar V-2 *Interface* inti GCF

*Interface* di atas diimplementasikan oleh kelas-kelas spesifik implementasi MIDP perangkat bergerak. Setiap vendor perangkat mobile mengimplementasikan *interface* tersebut secara berbeda. Berikut gambar yang menunjukkan *interface* GCF dan kelas-kelasnya.



**Gambar V-3** Interface dan kelas GCF

Interface GCF juga mendukung *subset TCP/IP*, *Wireless Application Protocol* (WAP) dan *socket*. Kelas yang memegang peranan sentral pada paket javax.microedition.io adalah kelas Connector. Kelas ini bertanggung jawab sebagai *intrepeter URL*, *instantiator* kelas, dan mengembalikan *interface* yang tepat ke pemanggilnya.

Interface GCF menyediakan tingkat fungsionalitas yang beragam. Ketika menulis aplikasi perlu ditentukan *interface* mana yang dibutuhkan aplikasi. Sebagai contoh, jika aplikasi hanya memerlukan kemampuan untuk membuat *output stream* saja tanpa memerlukan pembacaan *input*, maka *interface* OutputConnection merupakan *interface* yang tepat untuk digunakan. Tabel berikut ini menunjukkan daftar *interface* GCF yang didefinisikan CLDC.

**Tabel V-1** Daftar *Interface* GCF

Interface	Deskripsi
Connection	Merupakan kelas dasar untuk semua <i>interface</i> GCF dan mendefinisikan <i>method</i> close ()
DatagramConnection	Menangani <i>input</i> dan <i>output</i> (I/O) berbasis paket
InputConnection	Menangani koneksi <i>input stream</i>
OutputConnection	Menangani koneksi <i>output stream</i>
StreamConnectionNotifier	Menangani koneksi <i>inputstream</i> yang asinkron
StreamConnection	Mengkombinasikan koneksi I/O ke dalam satu <i>interface</i>
ContentConnection	Menangani sebuah koneksi yang dapat menentukan informasi tipe <i>content</i>

GCF didefinisikan sebagai bagian dari konfigurasi CLDC. Namun, agar lebih fleksibel, CLDC tidak mendefinisikan protokol yang diperlukan oleh *profile*. Berdasarkan keinginan para pengembang, setidaknya MIDP harus mendukung protokol HTTP dan HTTPS.

Perusahaan pembuat perangkat bergerak dapat menambahkan protokol yang sesuai dengan kebutuhannya. Saat ini perangkat yang mendukung MIDP sudah mendukung protokol berikut ini. a. *Datagram*

- b. Komunikasi serial
- c. *Short Message Service (SMS)*
- d. *Multimedia Message Service (MMS)*
- e. *Cell Broadcast Service (CBS)*
- f. *Socket*
- g. *Secure Socket Layer (SSL)*
- h. *Server Socket*

Untuk mengetahui protokol yang didukung oleh suatu perangkat bergerak beserta kapabilitasnya, pengembang harus membaca dokumentasi perangkat tersebut.

### V.3 Connector

Kelas `Connector` merupakan inti dari GCF. Kelas `Connector` merupakan kelas *factory* yang digunakan untuk membuat koneksi ke suatu jaringan. Koneksi yang dikembalikan dari berbagai *static method* disebut *polymorphic* karena obyek yang dikembalikan dapat di-*casting* ke beberapa *interface* yang beragam.

Kelas `Connector` digunakan untuk hal berikut :

- a. Koneksi ke suatu jaringan dan mendapatkan informasi tentang jaringan tersebut.
- b. Mengembalikan obyek `InputStream` sehingga aplikasi dapat segera mulai menerima data.
- c. Mengembalikan obyek `OutputStream` sehingga aplikasi dapat segera mulai mengirim data.

Salah satu dari koneksi-koneksi tersebut dapat mendukung beberapa protokol, salah satu protokol yang harus didukung oleh koneksi tersebut adalah protokol HTTP atau HTTPS. *Vendor* perangkat *mobile* dapat menambah protokol tambahan yang menurut mereka sangat diperlukan. Sebagai contoh, beberapa perusahaan pembuat perangkat bergerak mengijinkan pembentukan koneksi `socket` atau menggunakan *interface* GCF untuk koneksi melalui *port* infra merah. Namun, penggunaan protokol tersebut dalam sebuah aplikasi akan menyebabkan aplikasi tersebut tidak fleksibel dan mungkin terbatas pada perangkat yang mendukung protokol tersebut.

Untuk menyediakan berbagai tingkatan dukungan protokol, *interface* GCF memberikan format URL untuk menentukan protokol apa yang akan digunakan untuk sebuah koneksi.

Kelas `Connector` merupakan kelas berbasis pada *factory design pattern*. Informasi yang relevan tentang konstruksi sebuah obyek dikirim ke *factory*. Berdasarkan informasi tersebut, *factory* mengembalikan sebuah obyek yang diperlukan dalam aplikasi dan informasi yang diperlukan GCF adalah URI. Untuk membuat obyek `Connection`, digunakan satu dari berbagai bentuk metod `open()`. Setelah koneksi dibuka dapat dibentuk *stream* yang didefinisikan dalam paket `java.io` untuk mengirim atau menerima *content*. Tipe *interface* `Connection` yang dikembalikan bergantung dari tipe konesinya. Contoh berikut ini menunjukkan satu bentuk method `open()`.

```
public static Connection open(String url, int mode, boolean
                             timeouts) throws IOException
```

Tabel V-2 menunjukkan parameter yang dapat diisikan pada *method* `Connector.open()`. **Tabel**

**V-2** Parameter dalam `Connector.open()`

Parameter	Deskripsi
<code>String url</code>	URL
<code>int mode</code>	Mode akses, seperti <code>READ</code> , <code>WRITE</code> dan <code>READ_WRITE</code>
<code>boolean timeouts</code>	Sebuah <i>flag</i> yang memberitahu bahwa koneksi perlu mengaktifkan eksepsi <i>time-out</i>

*Method* `open()` merupakan metod yang di-overload dan tidak semua parameter di atas selamanya diperlukan, bergantung kebutuhan. Contoh berikut ini menunjukkan penggunaan *method* `open()` untuk beberapa tingkat aksesibilitas, *mode*, dan nilai *time-out*.

- a. Membaca/menulis koneksi menggunakan HTTP

```
Connector.open("http://www.jcc-itb.com/myData");
```

- b. Membaca koneksi menggunakan HTTPS

```
Connector.open("https://www.jcc.com/Data", Connector.READ);
```

- c. Koneksi input menggunakan HTTP

```
Connector.openInputStream("http://www.jcc.com/data.txt");
```

Kelas `Connector` mengembalikan obyek `Connection`. Obyek yang dikembalikan tersebut dapat *dicasting* ke beberapa tipe *interface* yang didefinisikan GCF. Jika *casting* ke *interface* yang tipe protokolnya tidak *valid*, maka sebuah *exception* akan terjadi.

Informasi tentang koneksi yang terbentuk bisa didapatkan, namun GCF tidak dapat digunakan untuk mengirim dan menerima data tanpa membentuk obyek *stream*. Untuk itu, digunakan *interface* `Connection` yang mendefinisikan *method-method* untuk mengakses *interface* `InputStream` dan `OutputStream`. Penggalan kode program berikut ini menunjukkan penggunaan kelas yang didefinisikan

GCF dan membuat sebuah koneksi, menerima informasi, dan menutup koneksi menggunakan standar *stream I/O*.

**Listing V-1** Contoh penggunaan GCF untuk membentuk koneksi

---

```
import javax.microedition.io.*; import
java.io.*;
...
try{
    InputConnection in = (InputConnection)Connector.open(
        "http://www.jcc.com/data.txt",Connector.READ);
    InputStream input = in.OpenInputStream();
    StringBuffer data = new StringBuffer();
    char ch;

    while((ch=input.read()) != -1){
        data.append((char)ch);
    }

    input.close();
in.close();
} catch(IOException ioe){
    //tangani error di sini
}

```

---

...

---

#### V.4 Format *Universal Resource Locator (URL)*

Tipe koneksi yang dibentuk oleh kelas *Connector* berbasis pada URL yang dikirim ke suatu *method* terkait. URL *interface* GCF mengacu pada *Uniform Resource Identifiers (URI)* yang didefinisikan dalam RFC2396 dengan format sebagai berikut. Penjelasan lebih rinci tentang parameter dari format URL dijelaskan pada Tabel 7.2.

scheme://userinfo@host:port/url-path;parameter

**Tabel V-3** Parameter dalam Format URL

Parameter	Deskripsi
<b>scheme</b>	Protokol yang digunakan untuk koneksi
<b>user</b>	nama pengguna dan password ( <i>optional</i> karena alasan keamanan tidak direkomendasikan menulis password di url)
<b>host</b>	Nama <i>host</i> ( <i>fully qualified host name</i> ) atau alamat IP
<b>port</b>	Nomor <i>port</i> yang digunakan ( <i>optional</i> )
<b>url-path</b>	<i>Path</i> dari sumber yang diminta

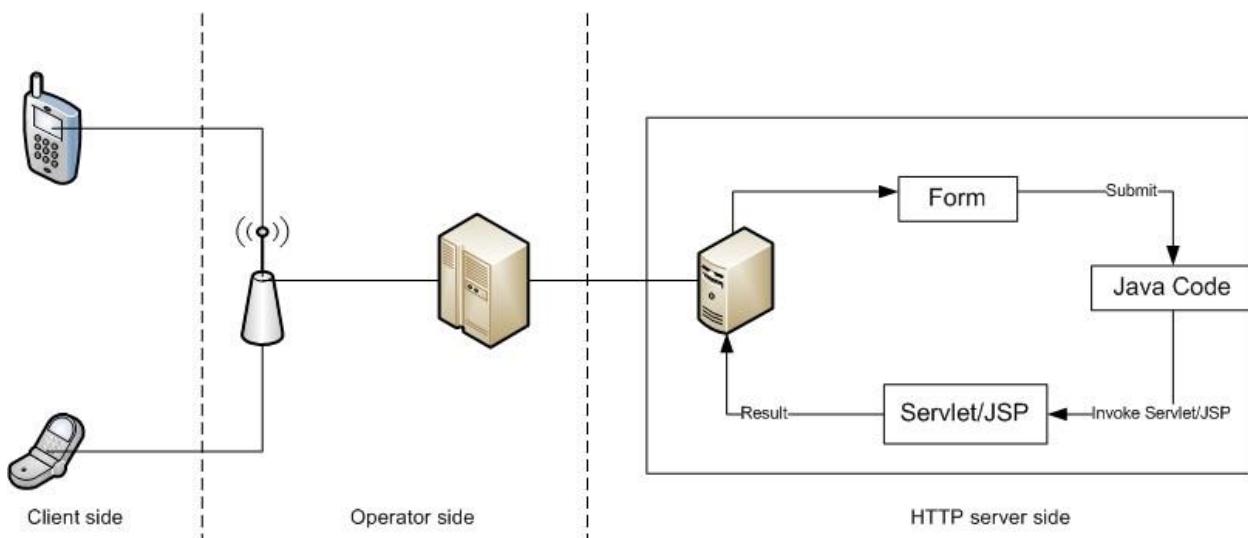
<b>parameter</b>	Informasi yang dikirim bersama <i>request (optional)</i> . Informasi ini didahului dengan titik koma
------------------	--

Perusahaan pembuat perangkat *mobile* dapat menambahkan tipe koneksi lain sesuai dengan kebutuhannya. Berikut ini contoh tipe koneksi beserta URL-nya.

- a. HTTP `http://www.jcc-itb.com/data`
- b. HTTPS `https://www.jcc-itb.com/dataRahasia`
- c. Serial I/O  
`comm://0:baudrate=9600`
- d. Datagram `datagram://192.168.0.22`

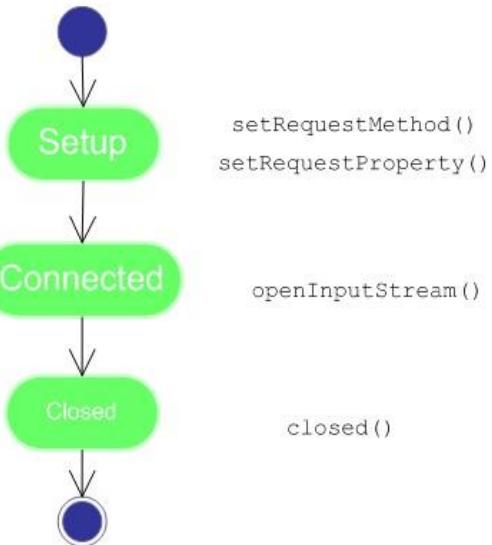
## V.5 Koneksi HTTP

Beberapa jenis aplikasi *mobile* memerlukan konektivitas HTTP dan HTTPS. Jika perangkat bergerak tidak memiliki dukungan TCP/IP internal maka konektivitas tersebut disediakan operator melalui sebuah *gateway*. Mekanisme ini dipakai luas pada perangkat *mobile* yang tersedia di pasaran. Gambar V-4 menunjukkan aplikasi *client-server* menggunakan HTTP.



**Gambar V-4** Arsitektur HTTP *client-server*

Sebelum membaca data, informasi tambahan yang didefinisikan dalam spesifikasi HTTP 1.1 dapat diperoleh. Untuk menemukan informasi ini terlebih dahulu perlu dilihat status/tahapan yang tercakup dalam sebuah koneksi HTTP. Gambar V-5 menunjukkan status yang dimaksud.



**Gambar V-5** Status koneksi HTTP

**Tabel V-4** Status koneksi HTTP

Status	Deskripsi
Setup	Status ini memanggil <i>method</i> <code>setRequestMethod()</code> dan <code>setRequestProperty()</code> untuk membangun pesan <i>connection request</i>
Connected	Koneksi ke sumber jaringan sudah dilakukan dan perangkat sudah terhubung (aktif) ke jaringan. Perubahan pada <i>connection request</i> tidak dapat dilakukan di status ini.
Closed	Koneksi ke sumber jaringan tidak lagi tersedia atau ditutup.

**Tabel V-5** *Method-method* untuk status Setup dan Connected

Status	Method yang tersedia
Setup	<code>setRequestMethod()</code> <code>setRequestProperty</code>
Connected	<code>close()</code>
	<code>getCharacterEncoding()</code>
	<code>getHost()</code>
	<code>getFile()</code>
	<code>getPort()</code>
	<code>getURL()</code>

getRequestMethod()
getType()

*Interface ContentConnection* menyediakan akses ke informasi *HTTP header*.

*Method getLength()* digunakan untuk menentukan ukuran *buffer* yang dialokasikan, *method getType()* dan *getEncoding()* untuk menentukan tipe *content* dan penanganannya.

*Interface HttpConnection* merupakan *interface* yang didefinisikan di MIDP 2.0. *Interface* ini menyediakan *method* untuk melakukan koneksi HTTP sebagai berikut.

- Menyeting *request header* dan metod-metod *request* sebelum request dibuat.
- Mendapatkan *header* balasan.

Perubahan pada *request header* dan metod-metod terkait hanya relevan dilakukan di tahap *Setup*. Setelah status berada di *Connected*, perubahan nilai-nilai tersebut tidak dapat dilakukan.

Transisi antara tahap *Setup* dengan *Connected* terjadi ketika beberapa *stream* atau informasi respon diperoleh dari *server*. Transisi ini terjadi ketika sebuah *input* atau *output stream* diperoleh dari suatu koneksi. Transisi ini juga terjadi jika beberapa *request* untuk mendapatkan informasi *respon header* dibuat, biasanya dengan menggunakan *method* *getXXXX* yang telah disediakan oleh *interface* *HttpConnection*.

HTTP *request header*, seperti *Accept: \*/\** atau *User Agent*, dapat diset. Namun perlu berhati-hati ketika menggunakan *request header* *Accept: \*/\** karena akan menyebabkan perangkat mendownload terlalu banyak informasi. Penggalan kode berikut ini menunjukkan penggunaan *method* *setRequestProperty()*.

#### **Listing V-2** Contoh penggunaan *method* *setRequestProperty()*

---

```
HttpConnection conn = (HttpConnection) Connector.open(
    "http://www.jcc.com"); conn.setRequestProperty("User-Agent", "Profile/MIDP-
2,0
                                Configuration/CLDC-1.0);

//Transisi ke status Connected
InputStream input = conn.getInputStream(); ...
```

---

#### **Listing V-3** Cara pengambilan informasi *header*

---

```
int responseCode = conn.getResponseCode(); int
dataLength = conn.getLength();
```

```
if ((responseCode == HttpConnection.OK) &&
    (dataLength != -1) {
    byte[] buf = new byte[dateLength]
    input.read(buf, 0, dataLength);
...

```

---

HTTP *Request* memiliki dua tipe, yaitu tipe *request* GET dan POST. tipe *request* GET adalah tipe metod HTTP yang difokuskan untuk mendapatkan sebuah *resource* atau sumber. Sumber tersebut dapat berupa file teks atau *image*. Karena *request* tipe GET adalah *request* yang umum sering digunakan di internet, maka tipe *request* ini menjadi nilai default dari obyek `HTTPConnection`.

Ketika melakukan *request*, aplikasi tidak hanya memerlukan informasi berupa URL saja, tetapi informasi lain. Sebagai contoh, sebuah *search engine* mungkin memerlukan nama dan nilai berupa suatu *string*. Jika suatu sumber masih memerlukan beberapa informasi lagi, maka informasi tersebut dapat ditambahkan pada URL. Berikut salah satu contoh penggunaan *method open()* dengan parameter sebuah alamat *server* diikuti dengan parameter "search" dengan nilai "seno".

```
Connector.open("http://www.jcc-itb.com?search=seno");
```

*Request* GET merupakan tipe *request* yang paling umum digunakan, namun tipe request ini memiliki keterbatasan. Versi HTTP sebelumnya membatasi panjang URL hanya sampai 255 karakter. Namun demikian, keterbatasan ini telah dihilangkan dalam HTTP 1.1.

Karena adanya persyaratan parameter *encoding* dalam URL dan pembatasan panjang karakter, *Request* tipe GET bukanlah tipe *request* yang selamanya baik untuk digunakan. Apabila informasi harus dikirimkan sebagai bagian dari *request* dan informasi tersebut ukurannya besar, maka *method POST* adalah tipe yang cocok untuk digunakan.

Jika metod GET membatasi banyaknya informasi yang dapat disertakan melalui parameter, maka tidak demikian halnya dengan metod POST. *Method POST* digunakan untuk mengirimkan informasi ke sebuah *server* untuk diproses dimana parameternya dikirim ke dalam *body request*.

Dalam metod POST, data berukuran besar seperti nama dan nilainya disertakan di dalam isi pesan.

Berikut ini contoh pengiriman data berukuran besar dengan menggunakan metod POST.

#### **Listing V-4** Contoh pengiriman data dengan POST

```
.....
HttpConnection conn = Connector.open(
                    "http://www.jcc-itb.com/servlet/process");
conn.setRequestMethod(HttpConnection.POST)
// Send Parameters in the body
```

---

```
OutputStream Out = conn.getOutputStream();
String parameters = "search=seno"; byte []
data = parameters.getBytes();
out.write(data); ....
```

---

Berikut akan disajikan potongan kode program yang menggunakan `HttpConnection`.

**Listing V-5** Contoh penggunaan `HttpConnection` (`PostMIDlet.java`)

```
import java.io.*;
import javax.microedition.io.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class PostMIDlet extends MIDlet
implements CommandListener, Runnable {
private Display mDisplay;    private Form
mForm;

public PostMIDlet() {
    mForm = new Form("Connecting...");
    mForm.addCommand(new Command("Exit", Command.EXIT, 0));
mForm.setCommandListener(this);
}
public void startApp() {
    if (mDisplay == null) mDisplay = Display.getDisplay(this);
mDisplay.setCurrent(mForm);

    // Do network loading in a separate thread.
Thread t = new Thread(this);
    t.start();
}
public void pauseApp() {}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command c, Displayable s) {
if (c.getCommandType() == Command.EXIT)
notifyDestroyed();
}
public void run() {
HttpConnection hc = null;
DataInputStream in = null;
DataOutputStream out = null;

try {
    String message = "name=Tamim";
    String url = "http://localhost:8080/examples/servlet/PostServlet";
    //Sesuaikan nama URL dengan URL server yang digunakan
```

```

        hc = (HttpConnection)Connector.open(url);
hc.setRequestMethod(HttpConnection.POST);
hc.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
hc.setRequestProperty("Content-Length",
Integer.toString(message.length()));           out =
hc.openDataOutputStream();
out.write(message.getBytes());          in =
hc.openDataInputStream();           int length =
(int)hc.getLength();           byte[] data = new
byte[length];           in.read(data);
String response = new String(data);
StringItem stringItem = new StringItem(null, response);
mForm.append(stringItem);           mForm.setTitle("Received
Data");
}catch (IOException ioe) {
StringItem stringItem = new StringItem(null, ioe.toString());
mForm.append(stringItem);           mForm.setTitle("Done.");
}finally {
try {
if (out != null) out.close();
if (in != null) in.close();           if
(hc != null) hc.close();
}catch (IOException ioe) {}
}
}
}

```

---

## V.6 Koneksi Socket

HTTP merupakan protokol yang sering digunakan untuk mengambil *file* dan melakukan interaksi dengan suatu *back end* aplikasi *web*, tetapi untuk aplikasi tertentu misalnya untuk *multiplayer game*, perlu dipertimbangkan alternatif protokol lain yang pengiriman datanya lebih ringan. *Socket* merupakan protokol yang dapat dipertimbangkan untuk keperluan tersebut. Melalui *socket* pengguna dapat berkomunikasi dengan terminal lainnya secara dua arah. Dalam hal ini, pengguna dapat mengirim data dan menerima data.

Untuk komunikasi TCP/IP, MIDP 2.0 menggunakan *Socket*. *Socket* sudah ada sejak tahun 80-an. Pada saat itu socket dimasukkan ke sistem operasi Unix dalam format *Berkeley Socket Interface* sebagai suatu cara untuk berkomunikasi melalui jaringan dengan berbagai macam protokol.

Ada 2 tipe socket yaitu *stream* dan *datagram*. Setiap tipe tersebut menggunakan protocol berbeda untuk berkomunikasi. *Socket stream* menggunakan TCP yang merupakan protokol *Connection-oriented*. Koneksi tersebut bersifat terbuka dan dikelola oleh program untuk proses pengiriman dan penerimaan data. Koneksi ini akan tertutup apabila *server* atau *client* secara eksplisit menulis perintah untuk menutup *socket*.

Pada komunikasi *stream* data diterima berdasarkan urutan data ketika dikirim. Karena *stream* sifatnya kontinu, urutan/susunan data yang diterima dijamin sama dengan urutan pada saat dikirim. Namun, karena komunikasi di jaringan internet cukup komplek, maka terkadang paket yang dikirim melalui *socket* seringkali tidak sesuai dengan urutan pada saat pengirimannya. Akibatnya, aplikasi harus menunggu paket yang hilang. Oleh sebab itu, *stream* bukan cara yang tercepat untuk mengirim data melalui internet.

Tipe *socket* lainnya adalah *socket datagram*. *Datagram* menggunakan protokol UDP. UDP berorientasi pada *record* dan tidak berorientasi pada koneksi sebagaimana halnya *stream*. *Datagram* merupakan potongan data-data, bukan stream yang sifatnya *kontinu*. Hal ini berarti paket yang dikirim tidak selamanya diterima *server* sesuai dengan urutan pengirimannya. Beberapa paket dapat saja terduplikasi dan bahkan beberapa di antaranya tidak sampai sama sekali. Oleh sebab itu, UDP merupakan protokol yang tidak handal (*unreliable*). namun demikian protokol ini menawarkan kecepatan. Paket UDP membutuhkan lebih sedikit *overhead* dan tidak perlu harus sesuai urutan paket ketika pengirimannya untuk sampai ke tujuan. Oleh karena itu paket UDP dapat dikirim dan diterima jauh lebih cepat daripada data yang dikirim melalui *stream*. Pengiriman melalui UDP biasanya melibatkan pengiriman pesan jawaban, menyertakan sebuah *time stamp* di dalam data yang dikirim, dan menggunakan berbagai skema prediksi untuk mengetahui kemungkinan terjadinya *gap* di antara paket yang terkirim.

*Datagram* dan *socket stream* merupakan fitur baru dari MIDP 2.0 meskipun TCP/IP bukan sesuatu yang dispesifikasikan MIDP 2.0. Fitur ini ditawarkan sebagai fitur utama pada setiap *vendor J2ME*.

*Socket stream* dan *Datagram* dihasilkan melalui kelas *SocketConnection*. Sebagaimana halnya pada *StreamConnection*, kelas ini diperoleh dari kelas *factory Connector*. Pada kasus ini protokol yang digunakan dispesifikasikan dalam sebuah format URL yang dimasukkan ke dalam metod *open* milik kelas *Connector*, sebagai contoh, *socket://* untuk TCP atau *datagram://* untuk UDP. Berikut contoh penulisan untuk membuka koneksi menggunakan *socket* :

```
SocketConnection sc = (SocketConnection)Connector.open(  
    "socket://127.0.0.1:5000");
```

Setalah alamat IP, ditulis nomor *port* yang digunakan, dalam hal ini 5000. Jika nomor *port* ini tidak ditentukan, maka MIDP berusaha untuk melakukan koneksi ke alamat IP bersangkutan dengan nomor *port* yang acak. Jika *port* ditemukan, maka dihasilkan sebuah koneksi socket TCP ke suatu *host* bersangkutan.

Koneksi *socket* UDP sama halnya dengan TCP, koneksi UDP ditentukan melalui sebuah alamat URL sebagaimana ditunjukkan berikut :

```
SocketConnection sc = (SocketConnection)Connector.open(  
    "datagram://127.0.0.1:5000");
```

Perbedaan yang tampak disini adalah dalam hal penggunaan tipe protokol, yaitu *datagram://*. Koneksi *socket* maupun *datagram* menggunakan *interface* yang sama, yaitu melalui *SocketConnection*.

Kelas `SocketConnection` dapat digunakan untuk memperoleh informasi tentang sebuah koneksi. Sebagai contoh, jika ingin mengetahui alamat sebuah koneksi *socket* yang sedang dihubungi, gunakan *method* berikut.

```
String getAdress();
```

*Method* ini mengembalikan sebuah *string* alamat tujuan yang saat ini sedang dihubungi. Untuk mengetahui nomor *port* dari sebuah koneksi, gunakan metode `getPort`:

```
int getPort();
```

`SocketConnection` diturunkan dari `Connection`. Oleh sebab itu, untuk menulis dan membaca data dapat digunakan antamuka `InputConnection` dan `OutputConnection` sebagaimana pada koneksi HTTP. Perbedaannya dengan HTTP, data *socket* berupa data biner sedangkan HTTP berupa teks.

Untuk membaca dan menulis data melalui koneksi *socket*, gunakan *method* berikut ini.

- a. `InputStream is = sc.openInputStream(); // Baca data`
- b. `OutputStream os = sc.OpenOutputStream(); // Tulis data`

Berikut disajikan potongan kode program yang menggunakan koneksi *socket*.

#### **Listing V-6** Contoh penggunaan koneksi *socket*

---

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

public class Communicator implements Runnable{
private Thread commThread;      private String
server;      private String port;      private
InputStream is;      private OutputStream os;
private SocketConnection socket;      private
boolean abort;      private String message;

    public Communicator() {
abort = false;      is =
null;      os = null;
socket = null;
    }
    public void setConnection( String server, String port){
this.server = server;      this.port = port;
    }
    public void sendMessage(String msg) {
abort = false;      message = msg;
startThread();
    }

    private void startThread() {      if
(settingsOk()) {
callback.loading();      commThread
= new Thread(this);
commThread.start();
    }
```

```

        }else{
            callback.settingsMissing();
        }
    }

private boolean settingsOk(){
    return !(server.trim().equals("") || port.trim().equals(""));
}

public void abort(){
abort = true;
}

public void run(){
StringBuffer response = null;
byte[] bytes = null;

try{
    byte bytes = message.getBytes();

    //kirim data ke server
    os.write( bytes );
os.flush();
callback.loading();

    if (abort){
disconnect();                  return;
    }

    //terima data dari server
response = new StringBuffer();
int ch = is.read();
    while
(ch != -1){
        response.append((char)ch);
ch = is.read();
    }

    String data = response.toString();
    // Proses data respon di sini..
}catch(Exception e ){
    disconnect();
return
}
}

public void disconnect(){
try{
    if(is != null){
is.close();                  is
= null;                      }
if( os != null ){
os.close();                  os
= null;
    }
    if(socket != null){
socket.close();              socket
= null;
    }
}

```

```

        } catch( IOException e ){
e.printStackTrace();
}
}

private void connect() throws Exception{
if (socket == null){
    SocketConnection socket = (SocketConnection)
        Connector.open("socket://" + server + ":" + port);

    os = socket.openOutputStream();
is = socket.openInputStream();
}
}
}

```

---

Program di atas mengirim data ke server socket dan menerima respon dari server socket. Proses koneksi ke server socket, membuka *input stream* dan *output stream* dilakukan melalui metod `connect()`. Pengiriman pesan ke server dilakukan melalui metod `sendMessage()` yang juga memulai proses `Thread` untuk mulai mengirim pesan ke *server* dan menerima respon dari *server*.

## VI    Multimedia pada J2ME

---

### VI.1 Sekilas Multimedia pada J2ME

Perkembangan perangkat bergerak saat ini mengarah kepada komunikasi *multimedia*. Pengguna dapat mengirim atau menerima informasi yang tidak hanya berbasis pada teks atau audio saja, melainkan video. Beberapa perangkat bergerak saat ini sudah diperkaya dengan fitur kamera dan kemampuan *multimedia player*, seperti MP3 dan video.

Di J2ME, profil MMAPI (Mobile Media API) memperluas cakupan ke arah pengembangan mobile *multimedia*. MMAPI yang didefinisikan dalam JSR-135 menyediakan sekumpulan *interface* dan kelas yang mendukung pemutaran berbagai tipe media.

Di dalam MIDP 2.0 terdapat *subset* spesifikasi MMAPI yang dibatasi. MMAPI mendefinisikan kelas dan *interface* dalam *package* `javax.microedition.media` dan `javax.microedition.media.player`. *Package-package* ini menyediakan bagian yang lebih kecil dari API yang didefinisikan dalam JSR-135. Manfaat dari *subset* yang terbatas dari API ini adalah aplikasi yang dibangun pada MIDP 2.0 dapat dijalankan pada perangkat yang mendukung MMAPI sehingga memperbanyak jumlah perangkat yang mendukung aplikasi MIDP 2.0.

Manfaat lainnya dari paket media dalam MIDP 2.0 adalah bahwa media tersebut dapat dikirim atau diperoleh melalui suatu jaringan atau sumber lokal. Fitur ini memungkinkan untuk menciptakan aplikasi

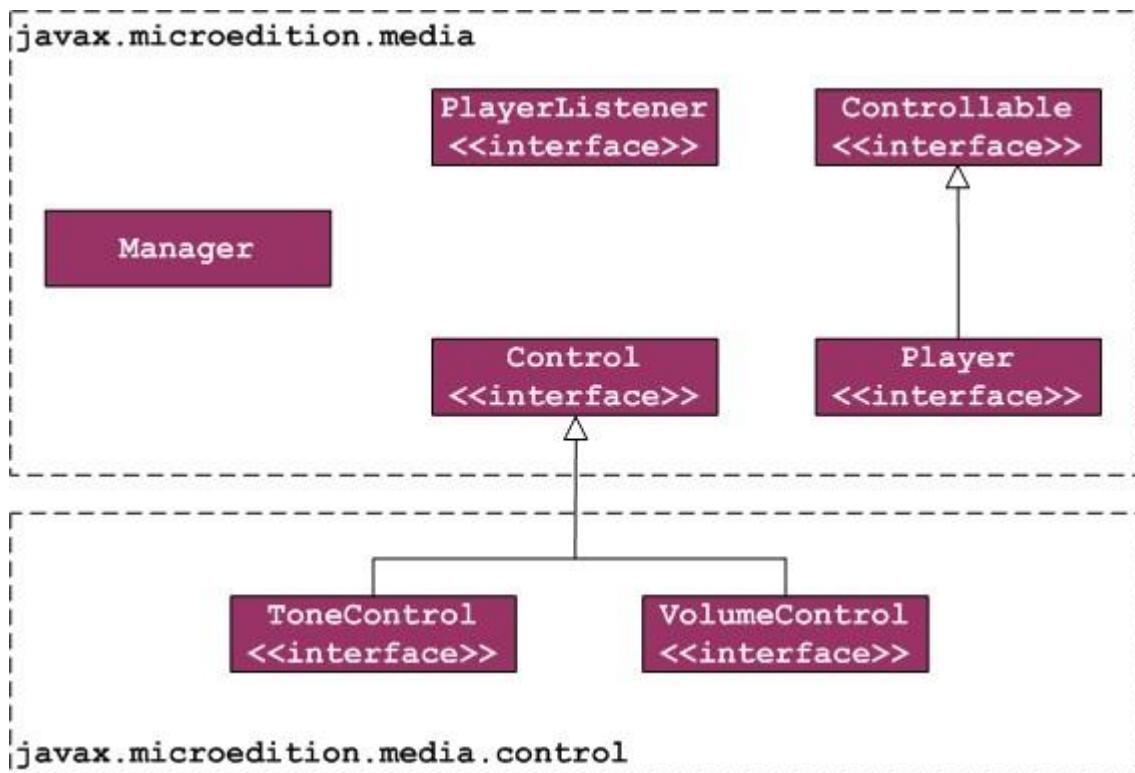
yang menarik seperti *game sound*, musik, dan *video clip*. Meskipun tidak semua fitur media didukung oleh MIDP 2.0, namun untuk menjaga kompatibilitas, fitur yang didukung haruslah memiliki panduan yang sama seperti yang ditentukan dalam JSR-135.

Perangkat bergerak saat ini sudah dapat mendukung berbagai tipe media. Namun, MIDP 2.0 menspesifikasikan bahwa tipe media setidaknya harus mendukung minimal hal berikut:

- a. Suara
- b. Modulasi *pulse-code* 8-bit (PCM) .wav dan .files

## VI.2 Mobile Media API

Paket MMAPI menyediakan *interface* untuk mendukung pemutaran berbagai media pada perangkat bergerak. Berikut gambar paket MMAPI pada Gambar VI-1.



Gambar VI-1 Package MMAPI

Paket MIDP 2.0 MMAPI memiliki 2 *package* yaitu :

- a. `javax.microedition.media.control` yang menyediakan *interface* untuk memainkan nada dering dan mengontrol *volume*.

- b. javax.microedition.media yang menyediakan kelas Manager. Kelas ini merupakan satu-satunya kelas konkret. Kelas Manager merupakan salah satu kelas Factory yang bertanggung jawab untuk :
  - i. Membuat obyek Player
  - ii. Melakukan *query* untuk mendapatkan informasi tentang protokol yang didukung oleh perangkat
  - iii. Melakukan *query* untuk mendapatkan informasi tentang tipe *content* yang didukung perangkat
  - iv. Memainkan *single tone*.

*Package* javax.microedition.media menyediakan *interface* berikut :

- a. *Interface* Control dalam paket javax.microedition.media.Control mendefinisikan tipe-tipe control yang berbeda, seperti *interface* VolumeControl.
- b. *Interface* Controllable digunakan untuk memperoleh sebuah control dan daftar suatu tipe kontrol.
- c. *Interface* Player digunakan untuk mengontrol siklus hidup *playback* dan komponen presentasi.
- d. *Interface* PlayerListener digunakan untuk menerima *asynchronous event* dari Player.

*Package* javax.microedition.media.Control menyediakan *interface* berikut ini.

- a. *Interface* ToneControl digunakan untuk memainkan (*playback*) urutan *tone*.
- b. *Interface* VolumeControl digunakan untuk mengatur volume pada sebuah obyek Player.

### VI.3 Manager dan Player

Terdapat dua kelas utama dalam hirarki MMAPI. Kelas tersebut adalah sebagai berikut:

- a. Kelas Manager merupakan salah satu Kelas Factory yang menyediakan *method* untuk mendapatkan obyek Player dan melakukan *query* terhadap protokol dan tipe *content* yang didukung oleh perangkat.
- b. Kelas Player menangani *playback* suatu *content*.

Untuk memperoleh obyek Player, panggil salah satu dari *method* *createPlayer()* berikut :

- a. Manager.createPlayer(InputStream stream, String type)
- b. Manager.createPlayer(String URL)

*Method* Manager.createPlayer (InputStream stream, String type) mengambil parameter berikut ini:

- a. Parameter `Stream` merepresentasikan `stream` dari `media` yang akan dimainkan. Jika lokasi `media` berada di penyimpanan lokal, metode `getResourcesAsStream()` dapat digunakan untuk memperoleh `stream` `media` tersebut. Jika lokasi `media` tidak berada di penyimpanan lokal, `stream` diperoleh dari suatu metode yang melakukan koneksi ke jaringan.
- b. Parameter `type` merepresentasikan tipe media suatu `stream`. Jika Parameter `type` bernilai `null`, maka kelas `Manager` menentukan tipe yang tepat. Jika Penentuan tipe `stream` ini tidak berhasil, maka akan dimunculkan eksepsi `MediaException`. Tabel berikut Menunjukkan beberapa tipe `media` yang umum.

**Tabel VI-1** Daftar tipe media

Tipe Media	Setting
<i>Wave .wav audio files</i>	Audio/x-way
<i>Access unit (AU) audio files</i>	Audio/basic
<i>MP3 audio file</i>	Audio/mpeg
<i>MIDI files</i>	Audio/midi
<i>Tone sequences</i>	Audio/x-tone-seq

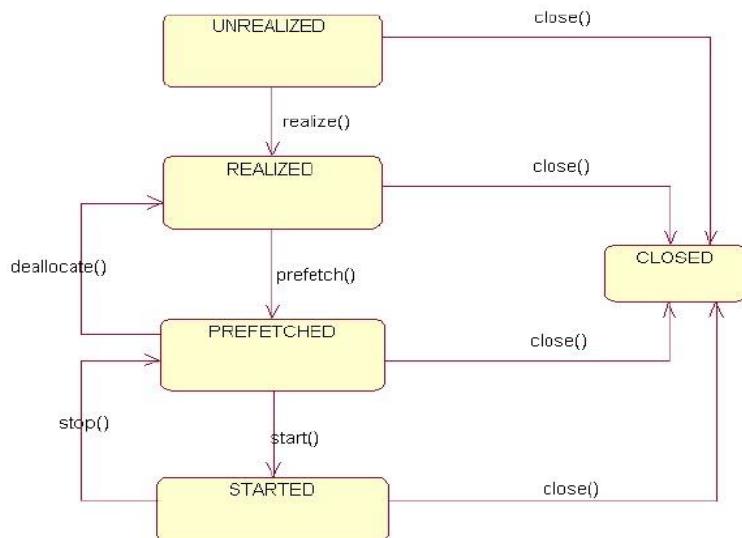
Kelas `Manager` menyediakan `method` untuk melakukan `query` dalam mendapatkan informasi tipe `content` yang didukung. `Query` ini dilakukan untuk mengetahui tipe `content` yang didukung oleh perangkat `mobile`.

`Method` `Manager.createPlayer(String URL)` menerima parameter URL. Parameter URL harus mengikuti sintaks URL umum. `Method` ini mencoba untuk mendapatkan `media` sesuai dengan informasi yang tertulis pada URL. Sebagai contoh, untuk mendapatkan sebuah file melalui HTTP, URLnya adalah sebagai berikut :

`http://www.somecoolsounds.com/takemeouttotheballgame.wav`

Sebagai catatan bahwa semua `method` yang didefinisikan oleh Kelas `Manager` merupakan `method` statik. Oleh sebab itu obyek `Manager` tidak perlu diinstansiasi.

Pada saat dijalankan, Obyek `Player` menjalani lima proses sebagaimana ditunjukkan pada gambar berikut.



**Gambar VI-2 Model Player state**

Pembahasan rinci mengenai tahapan di atas adalah sebagai berikut.

- UNREALIZED. Pada tahap ini obyek Player dibuat, namun obyek tersebut masih kosong dalam hal ini belum ada *media* yang diperoleh. Obyek Player ini dibentuk dari pengembalian *method createPlayer()*.
- REALIZED. Pada tahap ini lokasi sumber *media* sudah diidentifikasi. Proses ini memerlukan waktu beberapa saat. Untuk masuk ke tahap ini, *method Player.realized()* harus dipanggil.
- PREFETCHED. Pada tahap ini obyek Player sudah memiliki sumber *media* yang siap untuk dimainkan. Tahap ini belum mulai memainkan media, tetapi masih tahap pengisian *content* ke *buffer* obyek Player. Untuk masuk ke tahap ini, *method Player.prefetched()* harus dipanggil.
- STARTED – Pada tahap ini Obyek Player mulai memainkan media. Obyek Player kembali masuk ke tahap PREFETCHED jika *method stop()* dipanggil atau jika obyek Player telah mencapai akhir dari *media*. Sebuah obyek Player dapat menghentikan proses *rendering* sebelum akhir dari *media* tersebut dicapai. Untuk menjalankannya kembali dari suatu titik dimana ia dihentikan, *method Player.start()* harus dipanggil. Sebagai catatan bahwa pemanggilan *method start()* berakibat pada pemanggilan *method realize()* dan *prefetched()* secara implisit.
- CLOSED – Pada tahap ini, obyek Player telah melepaskan semua sumber media. Selanjutnya sumber media tersebut tidak dapat digunakan kembali. Untuk masuk ke tahap ini *method Player.close()* harus dipanggil.

Lima tahapan di atas sangat diperlukan mengingat proses *playback* memerlukan waktu. Tahapan di atas akan mengontrol pengoperasian *playback* dan juga akan mencegah konflik yang mungkin terjadi dari proses *playback*.

## VI.4 Media Audio

File audio yang dapat dijalankan menggunakan fitur multimedia pada J2ME dapat terdapat pada beberapa lokasi antara lain pada file yang terletak pada direktori `res` ataupun pada RMS. Pada potongan kode program berikut akan diperlihatkan kode untuk memainkan file audio dari dalam RMS.

**Listing VI-1** Memainkan audio dari RMS

---

```
RecordStore rs = RecordStore.open("name");
byte[] data = rs.getRecord(id);
ByteArrayInputStream is = new ByteArrayInputStream(data);
Player player = Manager.createPlayer(is, "audio/x-wav");

player.start();
```

---

Untuk memainkan *sound* yang sudah terpasang dalam suatu direktori di `midlet` biasanya di direktori `res`, lakukan beberapa hal berikut :

- a. Gunakan metod `getResourceAsStream()` milik obyek `Class`. *Method* ini mengembalikan sebuah *interface* `InputStream` yang digunakan obyek `Player` untuk membaca *media*.
- b. Tentukan tipe content, seperti `audio/x-wav`.
- c. Panggil metod `Manager.createPlayer(InputStream, type)` untuk membuat obyek `Player`
- d. Panggil metod `Player.play()` untuk memainkan *audio*.

Berikut ini penggalan kode program untuk memainkan sound yang terpasang dalam sebuah MIDlet JAR atau terletak pada direktori `res`.

**Listing VI-2** Memainkan audio dari direktori `res`

---

```
try(
    InputStream data = getClass().getResourcesAsStream("/explose.wav");
    String type = 'audio/x-wav';
    Player explode = Manager.createPlayer(data, type);
explode.start(); }catch(MediaException me) {
    //Tangani Error di sini
}catch(IOException ioe) {
    //Tangani Error lainnya di sini }
```

---

Berikut disajikan sebuah file `AudioPlayer.java` yang memainkan audio di perangkat bergerak. File berikut merupakan sebuah *thread* yang memainkan audio.

**Listing VI-3** File `AudioPlayer.java`

---

```
import javax.microedition.media.*;
import javax.microedition.media.control.*;
import java.io.*;

public class AudioPlayer implements Runnable {

    private MIDletController controller; private
    Player player;
    private VolumeControl volumeControl; private
    String url;

    public AudioPlayer(MIDletController controller){ this.controller
        = controller;
    }
    public void initializeAudio(String url){ this.url
        = url;
        Thread initializer = new Thread(this);
        initializer.start();
    }
    public void run(){
        try {
            player = Manager.createPlayer(url);
            controller.updateProgressGauge();
            player.addPlayerListener(controller);           player.realize();
            controller.updateProgressGauge();
            player.prefetch();
            controller.updateProgressGauge();
            volumeControl = (VolumeControl)player.getControl("VolumeControl");
            volumeControl.setLevel(50);           controller.updateProgressGauge();
            player.start(); }catch(IOException ioe) {
                controller.showAlert("Unable to connect to resource", ioe.getMessage());
            }catch (MediaException me) {
                controller.showAlert("Unable to create media player", me.getMessage());
            }
        }
    public void replay(){
        try{
            player.start();
        } catch (MediaException me) {
            controller.showAlert("MediaException thrown", me.getMessage());
        }
    }
    public void setVolume(int level){
        volumeControl.setLevel(level);
    }
}
```

```

public void close() {
player.close();
controller = null; url
= null;
}
}

```

---

Berikut disajikan MIDlet yang menggunakan *thread* diatas untuk memainkan audio.

**Listing VI-4 File MIDletController.java**

```

import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.media.*;
import javax.microedition.media.control.*;

public class MIDletController extends MIDlet implements
CommandListener, PlayerListener, ItemStateListener {

public MIDletController() {}
public void startApp(){ } public
void pauseApp(){ }
public void destroyApp(boolean unconditional){ } public
void commandAction(Command c, Displayable s){ }

public void playerUpdate(Player p, String event, Object eventData) {
if (event == PlayerListener.STARTED){ if(closeCommand == null){
closeCommand = new Command("close", Command.SCREEN, 1);
playerView.addCommand(closeCommand);
}
display.setCurrent(playerView);
} else if (event == PlayerListener.VOLUME_CHANGED) {
VolumeControl volumeControl = (VolumeControl)eventData;
int currentLevel = volumeControl.getLevel();
if(playerView.getVolumeIndicator() != currentLevel)
playerView.setVolumeIndicator(currentLevel);
} else if (event == PlayerListener.END_OF_MEDIA) {
if (replayCommand == null){
replayCommand = new Command("re-play", Command.SCREEN, 1);
}
playerView.addCommand(replayCommand);
}
}

public void itemStateChanged(Item item){ if
(item instanceof Gauge){
Gauge volumeIndicator = (Gauge)item;
int level = volumeIndicator.getValue();
audioPlayer.setVolume(level);
}
}

```

```
public void updateProgressGauge() {}  
public void showAlert(String title, String message) {}  
}
```

---

Kelas Manager memiliki *method static* berikut untuk menghasilkan nada dering sederhana :

```
static void playTone(int note, int duration, int volume)
```

Penjelasan mengenai parameter yang digunakan untuk *method* diatas dapat dilihat pada tabel VI-2 **Tabel**

**VI-2** Parameter *method* playTone ()

Parameter	Penjelasan
note	Nilai Nada yang dihasilkan harus berkisar antara nol hingga 127
duration	Jumlah waktu memainkan nada dering dalam milidetik
volume	Volume ketika memainkan nada dering : Nol untuk pengaturan tanpa <i>volume</i> Seratus untuk <i>volume</i> maksimum pada pengaturan perangkat yang ada saat ini.

Berikut disajikan potongan kode program yang menunjukkan cara menghasilkan sebuah nada dering sederhana.

**Listing VI-5** Memainkan nada dering sederhana

---

```
try{  
    Manager.playTone(65,1000,60);  
}catch(IllegalArgumentException iae){ }catch(MediaException  
me){ }
```

---

*Method* playTone () hanya dapat memainkan satu nada. Untuk memainkan nada yang beragam secara berurutan digunakan kelas Manager.

Urutan nada merupakan susunan *byte* [] yang memainkan *note* berdasarkan durasi, tempo dan versinya. Untuk memainkan urutan nada, perlu disusun suatu susunan *byte* [] yang berisikan elemen berikut ini :

a. Urutan durasi dan nada, dengan durasi berikut:

- i. 64 seluruh nada  
(default)
- ii. 32 setengah nada

- iii. 16 seperempat nada iv.  
seperdelapan nada
- b. Tempo yang hanya diatur sekali yaitu pada urutan awal. Dimulai dari 120 ketukan per menit (*beats per minute*)
- c. Versi yang harus diset 1

Sebagai catatan bahwa menyusun urutan nada adalah pekerjaan yang cukup rumit. Untuk lebih rinci, lihat dokumentasi API tentang *interface* TonePlayer.

Kelas Manager menyediakan konstanta TONE\_DEVICE\_LOCATOR. Untuk membuat sebuah obyek Player baru, kirim konstanta ini ke *method* createPlayer(String URL). Setelah obyek Player dikembalikan, tempatkan obyek Player ke tahap REALIZED.

*Package* javax.microedition.media.Control menyediakan sebuah *interface* ToneControl yang digunakan untuk memainkan urutan nada. Untuk memperoleh pengatur ini, panggil *method* getControl(String type) dan kirim string "ToneControl" ke argumen metod tersebut. Setelah urutan nada didefinisikan dan disimpan dalam suatu *array* byte[], panggil *method* berikut.

```
public void setSequence(byte[] sequence)
```

Penggalan kode program berikut menunjukkan cara memainkan suatu urutan nada.

#### **Listing VI-6** Memainkan suatu urutan nada

---

```
try{
    byte[ ] song = {ToneControl.VERSION,1,
                    67,16,69,16,67,8,65,8,64,48,62,8,60,
                    8,59,16,57,16,69,32,59,32};

    Player tonePlayer = Manager.createPlayer(Manager.TONE_DEVICE_LOCATOR);
    tonePlayer.realize();
    ToneControl tc = (ToneControl)tonePlayer.getControl("ToneControl");
    tc.setSequence(song);
    tonePlayer.start();
} catch(Exception e){
    //tangani eror do sini
}
```

---

## **VI.5 Media Video**

Memainkan video hampir sama dengan audio. Namun, perlu ditentukan tempat menampilkannya, di sebuah Form atau Canvas. Berikut penggalan kode program untuk memainkan video di atas Form :

#### **Listing VI-7** Contoh memainkan video diatas Form

---

```

InputStream is = getClass().getResourceAsStream("/somefile.3gp");
Player player = Manager.createPlayer(is, "video/3gpp"); player.realize();
VideoControl vc = (VideoControl)player.getControl("VideoControl"); if
(vc != null){
    Item it = (Item)vc.initDisplayMode(VideoControl.USE_GUI_PRIMITIVE,null);
myForm.append(it);
    p.start();
}

```

---

Berikut disajikan potongan kode program untuk memainkan *video* di atas Canvas.

**Listing VI-8** Contoh memainkan video diatas Canvas

---

```

VideoControl vc = (VideoControl)player.getControl("VideoControl"); if
(vc != null){
    vc.initDisplayMode(VideoControl.USE_DIRECT_VIDEO, myCanvas);
    vc.setVisible(true);      p.start();
}

```

---

Berikut disajikan contoh Canvas yang memainkan video.

**Listing VI-9** Contoh memainkan video diatas Canvas (*MovieCanvas.java*)

---

```

import javax.microedition.midlet.*; import
javax.microedition.lcdui.*; import
javax.microedition.media.*; import
javax.microedition.media.control.*; import
java.io.*;

public class MovieCanvas extends Canvas implements
                    CommandListener, PlayerListener, Runnable {

    private VideoPlayer parent;
    private Display display;
    private Player player;
    private VideoControl videoControl;
    private String url;      private
Thread initializer;      private Command
back;
    private Command rePlay;

    public MovieCanvas(VideoPlayer parent){
super();
    this.parent = parent;
    display = Display.getDisplay(parent);

```

---

```

        back = new Command("Back", Command.SCREEN, 1);
        addCommand(back);
        setCommandListener(this);
    }

    public void initializeVideo(String url){
        this.url = url;
        initializer = new Thread(this);
    initializer.start();
    }

    public void run(){
        try {
            player = Manager.createPlayer(url);
            player.addPlayerListener(this);
            player.realize();
            player.prefetch();
            } catch (IOException ioe) {
                Alert alert = new Alert("IOException thrown", ioe.getMessage(),
null, AlertType.ERROR); display.setCurrent(alert);
            } catch (MediaException me) {
                Alert alert = new Alert("MediaException thrown", me.getMessage(),
null, AlertType.ERROR); display.setCurrent(alert);
            }
            playVideo();
        }
    }

    public void playVideo(){
        try {
            // Ambil video control dan set video control ke display.
            videoControl = (VideoControl)player.getControl("VideoControl");
            if (videoControl != null) {
                videoControl.initDisplayMode(videoControl.USE_DIRECT_VIDEO,
                                              this);
            }
        }

        int cHeight = this.getHeight();
        int cWidth = this.getWidth();
        videoControl.setDisplaySize(cWidth, cHeight);
        display.setCurrent(this);
        videoControl.setVisible(true);           player.start();
        }catch (MediaException me) {
            Alert alert = new
            Alert("MediaException thrown",
                  me.getMessage(), null, AlertType.ERROR);
            display.setCurrent(alert);
        }
    }

    /** Tentukan warna background */
    public void paint(Graphics g){

```

```

        g.setColor(128, 128, 128);
        g.fillRect(0, 0, getWidth(), getHeight());
    }

    public void playerUpdate(Player p, String event, Object eventData) {
        //tambahkan opsi "Replay" ketika video is selesai diputar
        if (event == PlayerListener.END_OF_MEDIA){           if
(rePlay == null){
            rePlay = new Command("re-play", Command.SCREEN, 1);
            addCommand(rePlay);
        }
    }

    public void commandAction(Command c, Displayable s) {
        if(c == rePlay){
            try{
                player.start();
            } catch (MediaException me) {
                Alert alert = new Alert("MediaException thrown",
                        me.getMessage(), null, AlertType.ERROR);
                display.setCurrent(alert);
            }
        }
        else if(c == back){
            player.close();
            display.setCurrent(parent.mainForm);
            url=null;
            parent=null;
        }
    }
}

```

---

## VI.6 Capturing Media

MMAPI menyediakan dukungan untuk menangkap klip audio atau video dari *hardware* tertentu seperti *microphone* atau kamera dengan memanfaatkan sintaks “capture”.

Capture digunakan, sebagai contoh, untuk menampilkan *input* kamera ke layar. Jika perangkat *mobile* mendukung suatu RecordControl, maka implementasi tersebut dapat digunakan untuk merekam suatu *input media*.

Berikut disajikan potongan kode program untuk me-*capture* suatu audio.

**Listing VI-10** Contoh *capturing* audio

```

try {
    Player player = Manager.createPlayer("capture://audio");
    player.realize();
    RecordControl rc = (RecordControl)player.getControl("RecordControl");
    ByteArrayOutputStream output = new ByteArrayOutputStream();
    rc.setRecordStream(output);      player.prefetch();      rc.startRecord();
    player.start();
    Thread.currentThread().sleep(5000);
    rc.commit();      player.close(); } catch
    (IOException ioe) {
} catch (MediaException me) {
} catch (InterruptedException ie) {}

```

---

Program di atas merekam *audio* dari suatu perangkat *audio* selama 5 detik. Obyek Player dibuat menggunakan sintak "capture". Player harus diset ke status realized() sebelum sebuah kontrol diperoleh. Selanjutnya sebuah RecordControl diperoleh dari Player dan kontrol ini digunakan untuk merekam *audio* dari suatu perangkat *audio*. Method RecordControl startRecord() akan mulai merekam segera setelah Player mulai dijalankan. Jika Player belum dijalankan, recorder berada dalam kondisi *standby* sampai Player mulai dijalankan. Jika Player mulai dijalankan, maka aplikasi akan mulai merekam *audio*.

Berikut ini contoh penggalan kode program untuk mengambil gambar *snapshot* dari kamera.

**Listing VI-11** Contoh *capturing* gambar dari kamera

---

```

Player player;
VideoControl videoControl;
...
try{
    player = Manager.createPlayer("capture://video");
    player.realize();
    videoControl = (VideoControl)(player.getControl("VideoControl"));
    if (videoControl != null) {
        Form form = new Form("video");
        Item guiPrimitive = (Item)videoControl.initDisplayMode(
            videoControl.USE_GUI_PRIMITIVE, null);
        form.append(guiPrimitive);
        Display.getDisplay(midlet).setCurrent(form);
    }
    player.start(); }
catch(MediaException me){ }

```

---

Penggalan kode di atas menampilkan *video ouput* dari kamera. Untuk mendapatkan gambar *snapshot*, gunakan metod getSnapshot (...) milik obyek VideoControl.

**Listing VI-12** Contoh *capturing* video

---

```
try { byte[] pngImage = videoControl.getSnapshot(null);  
    // lakukan sesuatu dengan gambar  
} catch (MediaException me) { }
```

---

*Method* `getSnapshot(String imageType)` mengambil sebuah argumen `String` yang memberitahu format *image* yang akan dihasilkan. Nilai `null` pada argumen ini menghasilkan format *image default*, yaitu PNG. Untuk mengetahui format *image* yang didukung perangkat, gunakan *method* berikut ini.

```
System.getProperty(video.snapshot.encodings);
```

# VII Pustaka Game pada J2ME

---

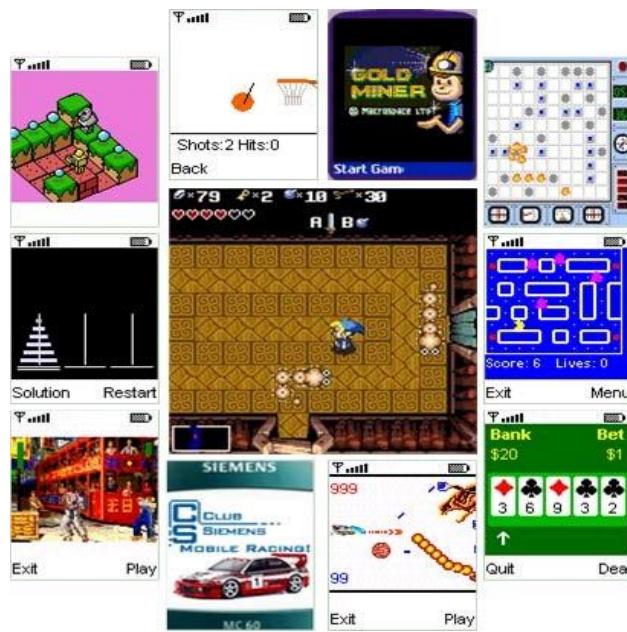
## VII.1 Sekilas Tentang Game

MIDP Game API merupakan pustaka J2ME yang sangat berguna di dalam pembuatan aplikasi game. Pustaka ini menyediakan semua fungsionalitas utama di dalam membuat aplikasi game dan memberikan berbagai kemudahan di dalam pengembangan game dalam konteks *mobile*. MIDP Game API menawarkan teknik-teknik pembuatan game yang efisien, terutama di dalam pembuatan konten game, yang berupa kemudahan di dalam mengkreasi gambar latar belakang dan gambar obyek animatif dengan menggunakan konsep *TiledLayer* dan *Sprite*.

Game adalah permainan yang sangat diminati masyarakat dan merupakan *value-added service* yang diandalkan penyedia konten layanan. Tipe game yang dapat dibuat sangat dibatasi oleh imajinasi manusia. Semakin kreatif imajinasi seseorang dan didukung dengan pengetahuan yang cukup dalam membuat game, maka sebuah konsep game baru dapat direalisasikan. Pada dasarnya, jenis-jenis game yang beredar di pasaran dapat dikelompokkan sebagai berikut.

- a. *Action game*
- b. *Adventure game*
- c. *Sport game*
- d. *Role playing game*
- e. *Platform game*
- f. *Puzzle game*

Jenis-jenis game ini dapat dimainkan oleh satu orang atau bermain dengan banyak orang dalam satu jaringan. Gambar berikut memperlihatkan game-game yang dibuat dengan J2ME.



**Gambar VII-1** Berbagai aplikasi game J2ME

## VII.2 Jenis-Jenis Game

Seperti disebutkan pada sub-bab sebelumnya terdapat enam jenis game. Berikut akan diulas masing-masing jenis game.

### a. Action game

*Action game* merupakan tipe game dengan fitur utama berupa fasilitas tembak cepat dengan banyak aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari tembakan musuh atau menghindari rintangan. Pengembang game tipe ini perlu memastikan game yang dibuat dioptimasi sehingga pemain memiliki pengalaman bermain yang baik, yang tidak terganggu oleh delay proses yang lama.

### b. Adventure game

*Adventure game* merupakan tipe game yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, mencari pesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat game ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita yang menarik bagi pemain.

### c. Sport game

*Sport game* merupakan tipe game yang berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau tim. Contoh game tipe ini antara lain sepakbola, bola basket, tenis, dan bilyard. Tergantung seberapa cepat permainan yang terjadi, aplikasi game perlu dioptimalkan.

### d. Role playing game

*Role playing game* merupakan tipe game yang seringkali berupa *multi-player game* di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang spesifik. Para pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafis yang khas untuk setiap karakter pemain akan sangat menarik dan memberikan pengalaman yang berbeda di dalam bermain.

e. *Platform game*

*Platform game* merupakan tipe game yang mengharuskan pemain mengarahkan suatu obyek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Tipe game ini sedikit serupa dengan *action game*, tetapi aksinya tidak secepat *action game*. Teknik *collision detection* sangat sering dimanfaatkan pada tipe game ini.

f. *Puzzle game*

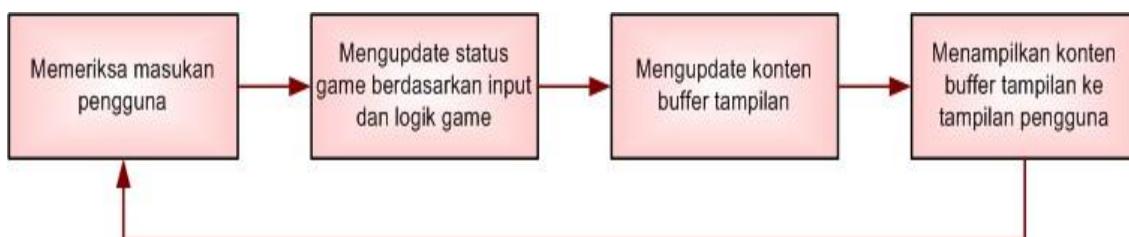
*Puzzle game* merupakan tipe game ini umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. Game ini lebih bersifat statis dibanding *action game*. Pembuatan game tipe ini seringkali ditunjang dengan algoritma *artificial intelligence*.

### VII.3 GameCanvas

Aplikasi game umumnya bersifat *time-driven*, di mana aplikasi terus berjalan dan tampilan terus diupdate ada atau tidaknya masukan dari pengguna. Kelas Canvas yang telah dipelajari didesain untuk aplikasi yang bersifat *event-driven*, di mana tampilan diupdate berdasarkan respon terhadap masukan pengguna.

Implementasi MIDP 2.0 menyediakan GameCanvas untuk mengatasi perbedaan ini. GameCanvas menyediakan sebuah model berbasis game untuk mengakses tampilan dan memeriksa masukan pengguna.

Umumnya aplikasi game mengimplementasikan sebuah *thread* untuk menjalankan game, yang secara kontinu memeriksa masukan pengguna, mengimplementasikan logik untuk mengupdate status game, dan mengupdate tampilan untuk merefleksikan perubahan status yang baru. Proses ini dapat dilihat pada gambar berikut.



Gambar VII-2 Aliran proses logik sebuah game

Agar lebih memahami karakteristik GameCanvas, berikut diberikan contoh bagaimana GameCanvas diterapkan di dalam membuat animasi sederhana, yang berupa gambar persegi empat yang semakin membesar. Pada contoh ini sama sekali tidak terdapat masukan dari pengguna.

---

**Listing VII-1** File AnimationGameCanvas.java

```
import javax.microedition.lcdui.Graphics; import
javax.microedition.lcdui.game.GameCanvas;

public class AnimationGameCanvas extends GameCanvas implements Runnable {

    private int w, h, stepX, stepY, index;

    public AnimationGameCanvas() {
super(true);      w = getWidth();
h = getHeight();      stepX =
w/20;      stepY = h/20;
    }
    protected void showNotify() {
new Thread(this).start();
    }
    public void run() {
        Graphics g = getGraphics();
        while(index<=10) {      try {
Thread.sleep(90);
        catch(InterruptedException e){}
index++;      render(g);
flushGraphics();
        }
    }
    public void render(Graphics g) {
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, w, h);

        g.setColor(0, 0, 64);
        g.fillRect(30, 30, w-60, h-60);

        g.setColor(255, 255, 255);
int incX = index * stepX;      int
incY = index * stepY;

        g.fillRect(0, 0, w/2 - incX, h);
        g.fillRect(w/2 + incX, 0, w/2 - incX, h);
        g.fillRect(0, 0, w, h/2 - incY);
        g.fillRect(0, h/2 + incY, w, h/2 - incY);
    }
}
```

---

## VII.4 Key Pooling

GameCanvas menyediakan kemampuan untuk memeriksa status *key* secara langsung (*polling*) daripada menunggu secara pasif *method callback* memproses setiap *key event*. Hal ini menarik bagi aplikasi game

karena memberikan kemampuan kontrol yang lebih terhadap aplikasi. *Key Pooling* ini melibatkan *key* yang didefinisikan sebagai *Game Action* pada Canvas.

GameCanvas memiliki sebuah *method* untuk mendapatkan status dari tiap *key*, yaitu *getKeyStates()*. Nilai integer kembalian *method* *getKeyStates()* menggunakan 1 bit untuk merepresentasikan setiap *game action* yang ada. Nilai bit 1 menandakan bahwa sebuah *key* ditekan, sedangkan nilai bit 0 berarti *key* tidak ditekan. Berikut adalah konstanta bit pada GameCanvas.

- a. UP\_PRESSED
- b. DOWN\_PRESSED
- c. LEFT\_PRESSED
- d. RIGHT\_PRESSED
- e. FIRE\_PRESSED
- f. GAME\_A\_PRESSED
- g. GAME\_B\_PRESSED
- h. GAME\_C\_PRESSED
- i. GAME\_D\_PRESSED

Untuk menentukan apakah suatu *key* ditekan, nilai integer kembalian *method* *getKeyStates()* dilakukan operasi AND dengan nilai kontanta di atas. Berikut diberikan contoh contoh penggunaan teknik *key polling*, yang hanya digunakan untuk menampilkan tulisan UP atau DOWN ketika *key* UP atau DOWN ditekan.

**Listing VII-2** File KeyPollingGameCanvas.java

---

```
import javax.microedition.lcdui.Graphics; import
javax.microedition.lcdui.game.GameCanvas;

public class KeyPollingGameCanvas extends GameCanvas implements Runnable {
private int w, h;    private String key;

    public KeyPollingGameCanvas() {
super(true);    w = getWidth();
h = getHeight();
}
protected void showNotify() {
new Thread(this).start();
}
public void run() {
    Graphics g = getGraphics();

    while(true) {        int ks =
getKeyStates();        if((ks &
UP_PRESSED) != 0)            key =
"UP";
        else if((ks & DOWN_PRESSED) != 0)
key = "DOWN";        else            key =
"";
    render(g);
flushGraphics();
}
```

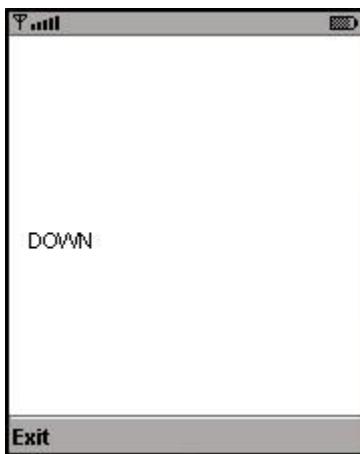
```

        try { Thread.sleep(90); }
catch(InterruptedException e){}
}
}

public void render(Graphics g) {
    g.setColor(0xffffffff);
    g.fillRect(0,0,w,h);
    g.setColor(0x000000);
    g.drawString(key, 10, h/2, Graphics.TOP|Graphics.LEFT);
}
}

```

---



**Gambar VII-3** Hasil program KeyPollingMidlet

## VII.5 Layer dan LayerManager

Layer adalah elemen grafik yang dapat dikombinasikan untuk membentuk sebuah tampilan yang lengkap. Teknik mengkombinasikan Layer-Layer merupakan analogi dari proses pembuatan animasi konvensional, di mana gambar latar belakang dan gambar utama digambar pada kertas transparan, yang kemudian gambar utama ditempatkan di atas gambar latar belakang untuk menghasilkan tampilan akhir.

Obyek dari kelas yang mewarisi `javax.microedition.lcdui.game.Layer` memiliki lokasi, ukuran, dan dapat dibuat terlihat ataupun tidak. Lokasi, ukuran, dan visibilitas Layer dapat diakses dan dimodifikasi menggunakan *method-method* berikut.

- public final int getX()
- public final int getY()
- public final int getWidth()
- public final int getHeight()
- public void setPosition(int x, int y)
- public final boolean isVisible()
- public void setVisible(boolean visible)

LayerManager berfungsi mengatur urutan sekumpulan Layer yang nantinya digunakan untuk membentuk suatu tampilan akhir yang diinginkan. Setiap Layer memiliki nilai indeks yang digunakan untuk menentukan posisi Layer dari yang paling depan hingga ke belakang. Indeks 0 berada paling atas atau paling dekat dengan pengguna. Urutan Layer ini seringkali dikenal dengan istilah “z order”.

Layer dapat ditambahkan ke bagian paling akhir dari daftar Layer atau ke lokasi yang spesifik menggunakan *method* berikut dari kelas LayerManager.

- a. public void append(Layer layer)
- b. public void insert(Layer layer, int index)

Untuk mendapatkan informasi jumlah Layer pada obyek LayerManager, *method* getSize() dapat digunakan.

*Method-method* berikut digunakan untuk mendapatkan Layer pada posisi tertentu dan menghapus Layer.

- a. public Layer getLayerAt(int index)
- b. public void remove(Layer layer)

LayerManager menggunakan konsep *view window* yang mengatur ukuran area yang ditampilkan dan posisinya terhadap sistem koordinat LayerManager. Perubahan posisi *view window* menyebabkan efek pergerakan tampilan bagi pengguna. Sebagai contoh, untuk menggeser tampilan ke kanan, posisi *view window* dapat dipindahkan ke kanan.

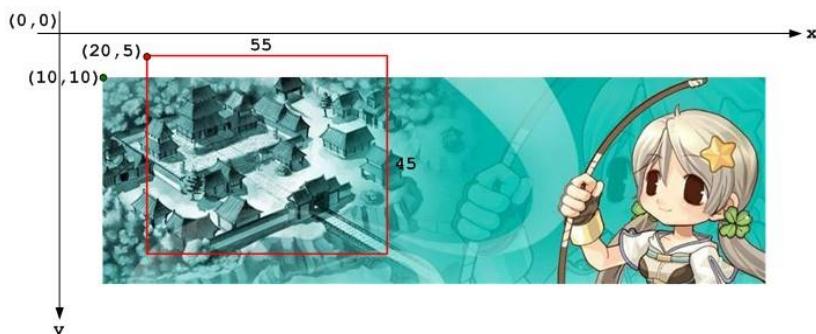
Dimensi *view window* mengatur seberapa besar tampilan yang dilihat pengguna. Dimensi ini umumnya diset sesuai dengan dimensi layar perangkat mobile. Untuk mengeset *view window* dengan nilai (x,y) relatif terhadap titik koordinat asal LayerManager, *method* berikut digunakan.

```
public void setViewWindow(int x, int y, int width, int height)
```

*Method* berikut digunakan untuk menampilkan posisi relatif *view window* terhadap layar tampilan.

```
public void paint(Graphics g, int x, int y)
```

Pada gambar berikut, *view window* ditempatkan di titik (20,5) pada sistem koordinat LayerManager dengan lebar dan tinggi sebesar 55 piksel.

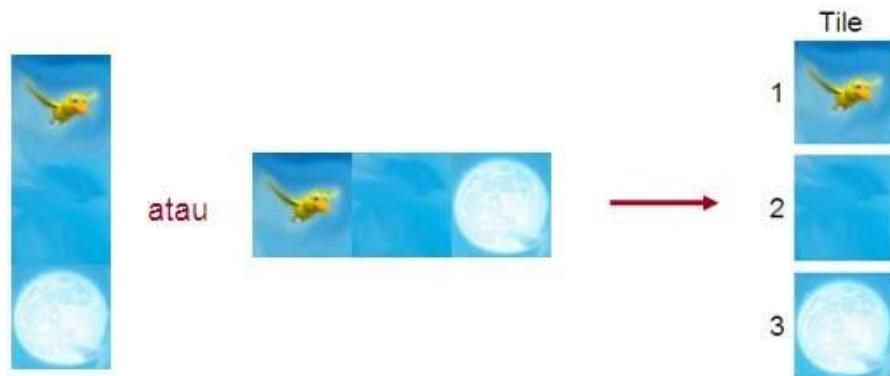


Gambar VII-4 Konsep *view window*

## VII.6 TiledLayer

TiledLayer merupakan elemen visual yang berupa area yang tersusun atas sel-sel berukuran sama, yang ditempati oleh satu set gambar *tile* (*tile image*). Kelas TiledLayer memungkinkan Layer virtual yang besar dibuat tanpa memerlukan gambar yang besar. Teknik ini umum digunakan di dalam pembuatan platform game 2D untuk membuat *scrolling background* yang besar.

*Tile*-*tile* yang digunakan untuk mengisi sel-sel TiledLayer disediakan oleh sebuah obyek *Image*, yang dibagi atas sekumpulan *tile* dengan ukuran yang sama. Seperti yang terlihat pada gambar berikut, satu set *tile* yang sama dapat disimpan dalam beberapa pengaturan.



Gambar VII-5 Set *tile* yang sama dapat disimpan dalam beberapa pengaturan

Setiap *tile* diberi indeks yang berbeda. *Tile* yang berada pada sisi kiri atas sebuah obyek *Image* diberi indeks 1, selebihnya diberi nomor sesuai dengan urutan baris. *Tile*-*tile* yang dihasilkan ini dinamakan *static tile* karena terdapat link yang tetap antara *tile* dan data obyek *Image* yang bersesuaian.

*Static tile* dibuat ketika melakukan instansiasi TiledLayer dan dapat diupdate kapan pun menggunakan *method* berikut.

```
public void setStaticTileSet(Image image, int tileSize, int tileHeight)
```

Selain *static tile*, pengembang aplikasi dapat mendefinisikan beberapa *animated tile*. *Animated tile* merupakan tile virtual yang secara dinamis bersesuaian dengan *static tile*. *Animated tile* memungkinkan pengembang aplikasi mengganti tampilan sebagian area dengan mudah. Dengan sebagian area diisi *animated tile*, maka tampilan area yang dinamis dapat dihasilkan dengan hanya mengganti *static tile* yang bersesuaian dengan *animated tile*. Teknik ini sangat berguna untuk membuat efek animasi berulang pada suatu area tampilan (seperti gambar pergerakan air dan pergerakan dedaunan).

*Animated tile* dibuat menggunakan *method* `createAnimatedTile (int staticTileIndex)` yang mengembalikan nilai indeks yang digunakan untuk *animated tile* yang baru dibuat. Indeks *animated tile* selalu bernilai negatif dan berurutan, yang dimulai dari -1. Ketika *animated tile* telah dibuat, *static tile* yang bersesuaian dengan *animated tile* dapat diganti dengan menggunakan *method* sebagai berikut.

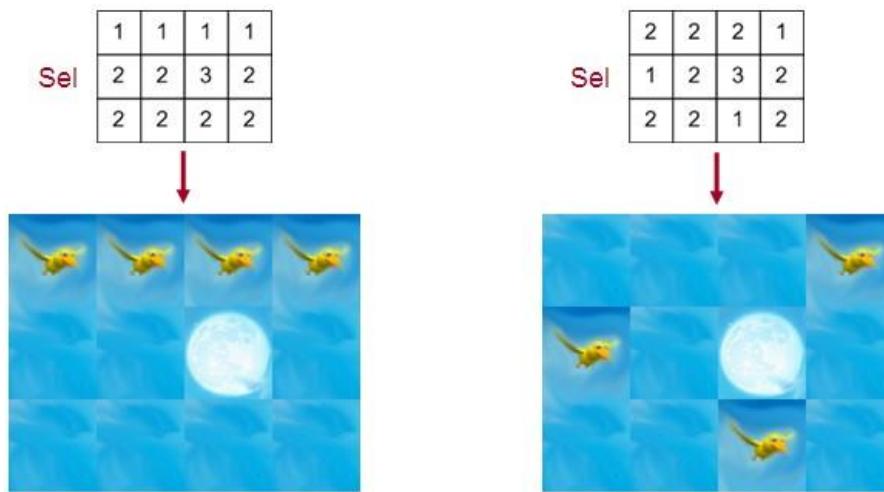
```
setAnimatedTile(int animatedTileIndex, int staticTileIndex)
```

Isi dari setiap sel penyusun *area grid* `TiledLayer` direferensi oleh nilai indeks *tile*. Nilai indeks positif merujuk ke *static tile*, sedangkan nilai indeks negatif merujuk ke *animated tile*. Nilai indeks 0 mengindikasikan sel yang bersangkutan kosong (transparan), tidak ditempati oleh tile.

Isi sebuah sel dapat diganti menggunakan *method-method* berikut.

- public void setCell(int col, int row, int tileIndex)
- public void fillCells(int col, int row, int numCols, int numRows, int tileIndex)

Contoh berikut mengilustrasikan bagaimana sebuah gambar dihasilkan menggunakan `TiledLayer`.



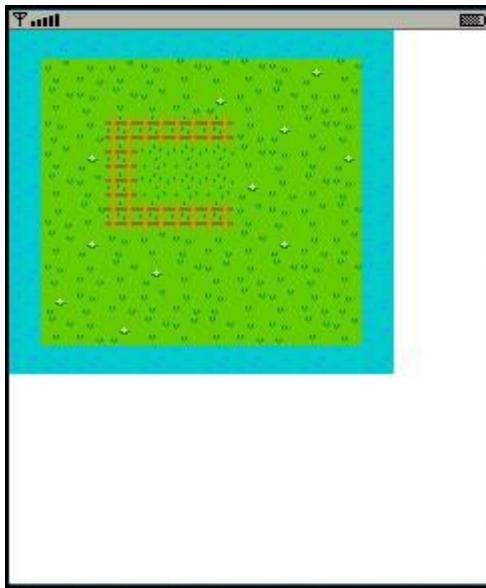
**Gambar VII-6** Grid sel merepresentasikan tile-tile penyusun gambar

Listing VI-3 sampai VI-5 merupakan contoh kode program untuk membuat sebuah gambar latar belakang menggunakan `TiledLayer` dan `LayerManager`. Program ini terdiri dari 3 buah file java, yaitu `TiledField.java`, `TiledGameCanvas.java` dan `TiledMidlet.java`.

Gambar sumber yang digunakan untuk menghasilkan sekumpulan tile dan hasil akhir gambar latar belakang yang dihasilkan dari Listing VI-3 sampai VI-5 dapat dilihat pada gambar berikut.



**Gambar VII-7** Sumber TiledLayer : Field.png (160 x 16 Piksel)



**Gambar VII-8** Latar belakang yang hendak dibuat

**Listing VII-3** File TiledField.java

---

```
import java.io.IOException; import
javax.microedition.lcdui.Image; import
javax.microedition.lcdui.game.TiledLayer;

public class TiledField extends TiledLayer {

    private static int[][] cellTiles = {
        {-3, -2, -3, -1, -2, -1, -3, -1, -2, -3, -1, -2},
        {-2, 3, 4, 3, 1, 2, 3, 2, 1, 5, 2, -3},
        {-1, 2, 1, 2, 3, 4, 5, 3, 2, 4, 3, -1},
        {-2, 1, 4, 9, 9, 9, 4, 5, 2, 1, -2},
        {-3, 3, 5, 9, 10, 10, 10, 2, 1, 3, 5, -1},
        {-2, 2, 3, 9, 10, 10, 10, 5, 4, 2, 1, -3},
        {-1, 4, 2, 9, 9, 9, 3, 1, 3, 2, -2},
        {-3, 2, 5, 1, 3, 1, 4, 2, 5, 4, 3, -3},
        {-2, 1, 4, 2, 5, 2, 3, 4, 2, 1, 2, -1},
        {-1, 5, 1, 4, 3, 4, 1, 2, 3, 4, 1, -2},
        {-3, 2, 4, 5, 2, 3, 2, 4, 1, 2, 3, -3},
        {-2, -3, -2, -1, -2, -1, -3, -2, -1, -3, -1, -2}
    };
    private static int[][] waterFrame = {{6,7,8}, {7,8,6}, {8,6,7}};

    private static final int TILE_WIDTH = 16;
    private static final int TILE_HEIGHT = 16;    private
    static final int WIDTH_IN_TILES = 12;    private
    static final int HEIGHT_IN_TILES = 12;

    private int tickCount = 0;

    public TiledField() {
```

```

        super(WIDTH_IN_TILES, HEIGHT_IN_TILES, createImage("/field.png"),
TILE_WIDTH, TILE_HEIGHT);      createAnimatedTile(waterFrame[0][0]); // tile
-1      createAnimatedTile(waterFrame[1][0]); // tile -2
createAnimatedTile(waterFrame[2][0]); // tile -3

        for(int row = 0; row < HEIGHT_IN_TILES; ++row) {
for(int column = 0; column < WIDTH_IN_TILES; ++column) {
setCell(column, row, cellTiles[row][column]);
}
}
}

public void tick() {
    int tickState = (tickCount++ >> 3);
int tile = tickState % 3;
    setAnimatedTile(-1-tile, waterFrame[tile][(tickState%9)/3]);
}
private static Image createImage(String filename) {
    Image image = null;
try {
    image = Image.createImage(filename);
} catch (IOException ex) { ex.printStackTrace(); }
return image;
}
}

}

```

---

**Listing VII-4** File TiledGameCanvas.java

---

```

import javax.microedition.lcdui.Graphics; import
javax.microedition.lcdui.game.GameCanvas; import
javax.microedition.lcdui.game.LayerManager;

public class TiledGameCanvas extends GameCanvas implements Runnable {

    private TiledField field;    private
LayerManager layerManager;    public
TiledGameCanvas() {    super(true);
setFullScreenMode(true);
layerManager = new LayerManager();
field      = new TiledField();
layerManager.append(field);
}
protected void showNotify() {
new Thread(this).start();
}
public void run() {    Graphics g =
getGraphics();    while(true) {
field.tick();    render(g);
flushGraphics();    try {
Thread.sleep(25); }
catch(InterruptedException e){}}

```

```

        }
    }

    public void render(Graphics g) {
        g.setColor(255, 255, 255);
        g.fillRect(0, 0, getWidth(), getHeight());
        layerManager.paint(g, 0, 0);
    }
}

```

---

**Listing VII-5 File TiledMidlet.java**

```

import javax.microedition.lcdui.Command; import
javax.microedition.lcdui.CommandListener; import
javax.microedition.lcdui.Display; import
javax.microedition.lcdui.Displayable; import
javax.microedition.midlet;

public class TiledMidlet extends MIDlet {

    protected void startApp() {
        TiledGameCanvas canvas = new
        TiledGameCanvas();
        canvas.addCommand(new
        Command("Exit", Command.EXIT,
        0));
        canvas.setCommandListener(new
        CommandListener() {
            public void
            commandAction(Command c,
            Displayable d) {
                notifyDestroyed();
            }
        });
        Display.getDisplay(this).setCurrent(canvas);
    }
    public void destroyApp(boolean unconditional) {}

    public void pauseApp() {}
}

```

---

## VII.7 Sprite

Sprite merupakan elemen visual yang dapat dirender dengan sebuah frame dari frame-frame yang terdapat pada Image, di mana frame yang berbeda dapat ditampilkan untuk memberi efek animasi. Jika lebih dari sebuah frame yang digunakan, gambar sumber dapat dibagi atas beberapa frame dengan ukuran

yang sama. Pada Gambar 7.10, diperlihatkan bahwa satu set frame yang sama dapat disimpan dalam beberapa pengaturan.



**Gambar VII-9** Set frame yang sama dapat disimpan dalam beberapa pengaturan

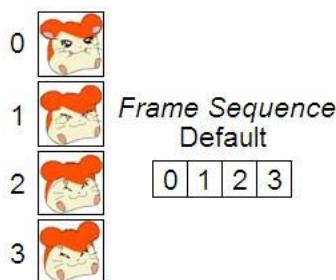
Setiap frame diberi indeks yang berbeda. Frame yang berada pada sisi kiri atas sebuah obyek `Image` diberi indeks 0, selebihnya diberi nomor sesuai dengan urutan baris. `Method int getRawFrameCount ()` digunakan untuk mendapatkan jumlah *raw frame* yang dihasilkan dari sebuah `Image`.

Untuk menentukan urutan tampilan frame, maka perlu didefinisikan *frame sequence*. *Frame sequence* default merujuk kepada daftar frame yang tersedia, di mana terdapat hubungan langsung antara indeks *sequence* dengan indeks frame, dan panjang *frame sequence* default sama dengan jumlah *raw frame*. Sebagai contoh, jika sebuah `Sprite` memiliki 4 frame, *frame sequence* defaultnya adalah {0, 1, 2, 3}, seperti yang diilustrasikan pada Gambar 7.11. *Frame sequence* juga dapat didefinisikan sendiri dengan merepresentasikannya sebagai *array of integer* selanjutnya memasukkannya ke dalam *method setFrameSequence(int[])*.

```
void setFrameSequence(int[] sequence)
```

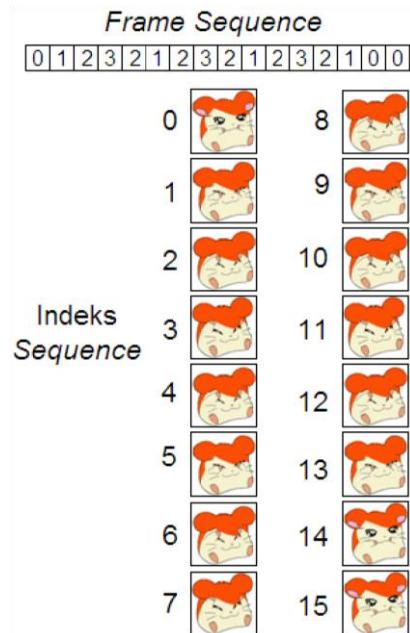
Pembuat game secara manual melakukan pergantian frame yang aktif pada *frame sequence* dengan memanggil salah satu dari *method-method* berikut.

- a. public void setFrame(int sequenceIndex)
- b. public void prevFrame()
- c. public void nextFrame()



**Gambar VII-10** Frame sequence default

Contoh berikut memperlihatkan bagaimana suatu *frame sequence* digunakan untuk membuat efek animasi sebuah gambar. *Frame sequence* didesain sehingga karakter hamtaro menggerakkan kedua tangannya naik turun pada wajahnya ketika *method* `nextFrame()` dipanggil setiap tampilan diupdate.



**Gambar VII-11** Pengaturan *frame sequence* untuk memberi efek animasi

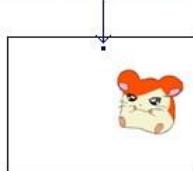
## VII.8 Reference Pixel

Sprite merupakan turunan dari `Layer`, yang mewarisi berbagai *method* untuk mengeset dan mendapatkan informasi lokasinya, seperti `setPosition(int x, int y)`, `getX()` dan `getY()`. Semua *method* ini mendefinisikan posisi terhadap titik koordinat yang berada pada sudut kiri atas batas visual `Sprite` sebagai titik referensinya. Namun, untuk beberapa kasus, akan memudahkan jika dapat mendefinisikan posisi `Sprite` terhadap sembarang titik koordinat di dalam frame sebagai titik referensinya, terutama jika proses transformasi dilakukan pada `Sprite`.

Terkait dengan keperluan di atas, `Sprite` menyertakan konsep *reference pixel*. *Reference pixel* didefinisikan dengan menentukan lokasinya pada *raw frame* menggunakan `void defineReferencePixel(int x, int y)`. Secara default, *reference pixel* adalah piksel yang berada pada titik koordinat (0, 0) suatu frame. *Reference pixel* juga dapat didefinisikan di luar batas frame.

Pada contoh berikut, *reference pixel* didefinisikan pada titik (50, 5).

```
defineReferencePixel(50,5)
```



**Gambar VII-12** Reference pixel

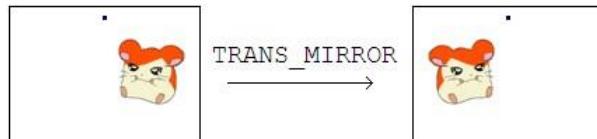
*Method* `getRefPixelX()` dan `getRefPixelY()` dapat digunakan untuk memperoleh informasi lokasi *reference pixel* *Sprite* pada sistem koordinat tampilan. Untuk mengganti posisi *Sprite* berdasarkan *reference pixel*, *method* `setRefPixelPosition(int x, int y)` dapat digunakan.

## VII.9 Transformasi Sprite

Transformasi pada *Sprite* dilakukan dengan menggunakan *method* `void setTransform(int transform)`. Transformasi pada *Sprite* pada dasarnya sama dengan transformasi pada *Image* biasa. Parameter transformasi juga sama dengan transformasi pada *Image*. Transformasi ini dapat bermanfaat mengurangi jumlah frame yang dibutuhkan untuk membuat animasi. Parameter transformasi yang dimungkinkan sebagai berikut.

- a. `TRANS_NONE`
- b. `TRANS_ROT90`
- c. `TRANS_ROT180`
- d. `TRANS_ROT270`
- e. `TRANS_MIRROR`
- f. `TRANS_MIRROR_ROT90`
- g. `TRANS_MIRROR_ROT180`
- h. `TRANS_MIRROR_ROT270`

Berikut adalah contoh transformasi yang dilakukan dengan parameter `TRANS_MIRROR`.



**Gambar VII-13** Transformasi dengan parameter `TRANS_MIRROR`

## VII.10 Collision Detection

Fungsionalitas *collision detection* merupakan fitur penting bagi aplikasi game. Tabrakan atau *collision* yang dimaksud adalah antara dua atau lebih *Sprite*, *TiledLayer* atau *Image* dalam satu tampilan. Terkait dengan ini, Game API mendukung dua teknik untuk mendeteksi terjadinya *collision*/tabrakan yaitu:

- a. Implementasi dapat membandingkan area-area persegi empat yang merepresentasi-kan sebuah *Sprite* dengan *Sprite*, *TiledLayer* atau *Image* lainnya. *Collision* terjadi jika area-area persegi empat tersebut beririsan. Teknik ini merupakan cara yang cepat untuk mencollision detection, tetapi memungkinkan terjadinya ketidaktepatan di dalam mencollision detection untuk bentuk-bentuk yang tidak persegi empat.
- b. Implementasi dapat membandingkan tiap piksel sebuah *Sprite* dengan *Sprite* lainnya. Jika sebuah piksel yang tidak transparan (*opaque*) dari sebuah *Sprite* beririsan dengan sebuah piksel yang tidak transparan (*opaque*) dari *Sprite* yang lain, *collision* terjadi. Teknik ini memerlukan komputasi yang lebih banyak, tetapi akan memberikan hasil yang lebih akurat.

*Sprite* memiliki *collision rectangle*, yang secara default terletak pada titik (0, 0) dengan lebar dan tinggi yang sama dengan obyek *Sprite*. Untuk mengganti *collision rectangle*, *method* berikut digunakan.

```
public void defineCollisionRectangle(int x, int y, int width, int height)
```

*Collision rectangle* memiliki dua peran sebagai berikut.

- a. Jika *collision detection* berdasarkan piksel (*pixel-level collision detection*) tidak digunakan, *collision rectangle* digunakan sebagai area untuk mendeteksi *collision*.
- b. Jika *collision detection* berdasarkan piksel (*pixel-level collision detection*) digunakan, maka hanya piksel di dalam *collision rectangle* yang diproses.

*Sprite* memiliki kemampuan untuk mencollision detection dengan *Sprite*, *TiledLayer* dan *Image*. Berikut adalah *method-method* yang merealisasikan kemampuan tersebut.

- a. public final boolean collidesWith(Sprite s, boolean pixelLevel)
- b. public final boolean collidesWith(TiledLayer t, boolean pixelLevel)
- c. public final boolean collidesWith(Image img, int x, int y, boolean pixelLevel)

## VII.11 Efek Spesial

Seringkali untuk membuat game semakin menarik, para pengembang menambahkan beberapa efek spesial, seperti memberikan efek getar pada perangkat mobile atau memberikan efek cahaya ketika sebuah game dimainkan. J2ME menyediakan *method-method* yang sering digunakan di dalam pembuatan game untuk memberikan efek spesial selama game tersebut dimainkan. *Method-method* yang dimaksud sebagai berikut.

- a. public boolean flashBacklight(int duration)
- b. public boolean vibrate(int duration)

Kedua *method* di atas didefinisikan pada kelas *Display*. *Method-method* tersebut menerima parameter durasi dalam milidetik yang digunakan untuk menentukan berapa lama *backlight* perlu dinyalakan dan berapa lama perangkat *mobile* perlu digetarkan. Kedua *method* tersebut mengembalikan nilai *true* jika berhasil dan *false* jika perangkat *mobile* yang digunakan tidak mendukung *backlight* atau *vibration*.

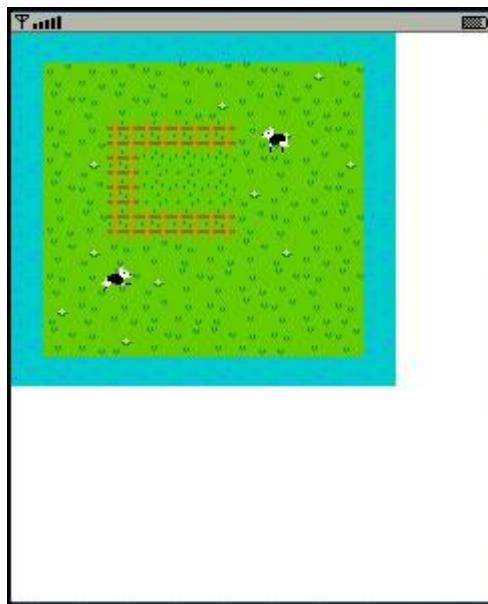
## VII.12 Pemakaian *Sprite*, *Collision Detection* dan Efek Spesial dalam Aplikasi

Pada bagian ini diberikan contoh pemakaian *Sprite*, *collision detection*, dan efek spesial di dalam sebuah aplikasi. Program yang disediakan terdiri dari 4 buah file java, yaitu *TiledField.java*, *SpriteDog.java*, *SpriteGameCanvas.java* dan *SpriteMidlet.java*.

Gambar sumber yang digunakan untuk menghasilkan sekumpulan frame dan hasil akhir aplikasi dari Listing VII-6 sampai VI-9 dapat dilihat pada gambar berikut.



**Gambar VII-14** Sumber *Sprite* : Dog.png (225 x 15 Piksel)



**Gambar VII-15** Hasil program yang menerapkan *Sprite*, *Collision Detection* dan Efek Spesial

---

### Listing VII-6 File *TiledField.java*

```

import java.io.IOException; import
javax.microedition.lcdui.Image; import
javax.microedition.lcdui.game.TiledLayer;

public class TiledField extends TiledLayer {

    private static int[][] cellTiles = {
        {-3, -2, -3, -1, -2, -1, -3, -1, -2, -3, -1, -2},
        {-2, 3, 4, 3, 1, 2, 3, 2, 1, 5, 2, -3},
        {-1, 2, 1, 2, 3, 4, 5, 3, 2, 4, 3, -1},
        {-2, 1, 4, 9, 9, 9, 9, 4, 5, 2, 1, -2},
        {-3, 3, 5, 9, 10, 10, 10, 2, 1, 3, 5, -1},
        {-2, 2, 3, 9, 10, 10, 10, 5, 4, 2, 1, -3},
        {-1, 4, 2, 9, 9, 9, 9, 3, 1, 3, 2, -2},
        {-3, 2, 5, 1, 3, 1, 4, 2, 5, 4, 3, -3},
        {-2, 1, 4, 2, 5, 2, 3, 4, 2, 1, 2, -1},
        {-1, 5, 1, 4, 3, 4, 1, 2, 3, 4, 1, -2},
        {-3, 2, 4, 5, 2, 3, 2, 4, 1, 2, 3, -3},
        {-2, -3, -2, -1, -2, -1, -3, -2, -1, -3, -1, -2}
    };
    private static int[][] waterFrame = {{6,7,8}, {7,8,6}, {8,6,7}};

    private static final int FENCE_TILE = 9;
    private static final int TILE_WIDTH = 16;
    private static final int TILE_HEIGHT = 16;
    private static final int WIDTH_IN_TILES = 12;
    private static final int HEIGHT_IN_TILES = 12;
    private int tickCount = 0;

    public TiledField() {
        super(WIDTH_IN_TILES, HEIGHT_IN_TILES, createImage("/field.png"),
              TILE_WIDTH, TILE_HEIGHT);

        createAnimatedTile(waterFrame[0][0]); // tile -1
        createAnimatedTile(waterFrame[1][0]); // tile -2
        createAnimatedTile(waterFrame[2][0]); // tile -3

        for (int row = 0; row < HEIGHT_IN_TILES; ++row) {           for
            for (int column = 0; column < WIDTH_IN_TILES; ++column) {
                setCell(column, row, cellTiles[row][column]);
            }
        }
    }

    public void tick() {
        int tickState = (tickCount++ >> 3);
        int tile = tickState % 3;
        setAnimatedTile(-1-tile, waterFrame[tile][(tickState % 9) / 3]);
    }

    public boolean containImpassableArea(      int
        int x, int y, int width, int height) {      int rowMin
        = y / TILE_HEIGHT;      int rowMax = (y + height -
        1) / TILE_HEIGHT;
        int columnMin = x / TILE_WIDTH;
    }
}

```

```

int columnMax = (x + width - 1) / TILE_WIDTH;

for(int row=rowMin; row<=rowMax; row++) {
    for(int column=columnMin; column<=columnMax; column++) {
        int cell = getCell(column, row);
        if ((cell < 0) || (cell == FENCE_TILE)) {
            return true;
        }
    }
}
return false;
}

private static Image createImage(String filename) {
    Image image = null;
try {
    image = Image.createImage(filename);
} catch (IOException ex) {
    ex.printStackTrace();
}
return image;
}
}

```

---

**Listing VII-7 File SpriteDog.java**

```

import java.io.IOException; import
javax.microedition.lcdui.Image; import
javax.microedition.lcdui.game.Sprite;

public class SpriteDog extends Sprite {

    public static final int NONE      = -1;
    public static final int UP        = 0;    public
    static final int LEFT       = 1;    public static
    final int DOWN      = 2;    public static final
    int RIGHT       = 3;

    private static final int WIDTH   = 15;
    private static final int HEIGHT  = 15;

    private static final int VIBRATION = 200;

    private static final int STAND   = 0;
    private static final int RUN     = 1;

    private int[][][] animations = {{{0},           // stand up
{1, 2, 3, 4}},          // run up
                                {{5},           // stand left
{6, 7, 8, 9}},          // run left
                                {{10}},          // stand down

```

```

        {11, 12, 13, 14}}}; // run down

    private int animationTick      = 0;
    private int currentDirection = LEFT;

    private SpriteGameCanvas canvas;
    private TiledField field;

    public     SpriteDog(SpriteGameCanvas    canvas)    {
super(createImage("/dog.png"),           WIDTH,          HEIGHT);
defineCollisionRectangle(2,   2,   WIDTH-4,   WIDTH-4);
defineReferencePixel(WIDTH/2,           HEIGHT/2);
setStandingAnimation();      this.canvas = canvas;
this.field  = canvas.getField();
    }
    private static Image createImage(String filename) {
        Image image = null;
try {
    image = Image.createImage(filename);      }
catch (IOException ex) {
ex.printStackTrace();
}
        return image;
    }
    public void tick(int direction) {
animationTick++;

    switch (direction) {
case UP:
        currentDirection = direction;
        if((getY() > 0) &&
!field.containImpassableArea(           getX(),
getY()-1, getWidth(), 1) &&
moveSuccessfully(0,-1)) {
setRunningAnimation();
} else {
        SpriteMidlet.vibrate(VIBRATION);
setStandingAnimation();
        }           break;           case LEFT:
currentDirection = direction;
if((getX() > 0) &&
!field.containImpassableArea(           getX()-
1, getY(), 1, getHeight()) &&
moveSuccessfully(-1,0)) {
setRunningAnimation();
} else {
        SpriteMidlet.vibrate(VIBRATION);
setStandingAnimation();
        }           break;           case
DOWN:           currentDirection =
direction;
        if((getY() + getHeight() < field.getWidth()) &&
!field.containImpassableArea(
getX(), getY()+getHeight(), getWidth(), 1) &&

```

```

moveSuccessfully(0,1)) {
setRunningAnimation();
} else {
    SpriteMidlet.vibrate(VIBRATION);
setStandingAnimation();
    } break; case
RIGHT:         currentDirection =
direction;
if((getX() + getWidth() < field.getWidth()) &&
    !field.containsImpassableArea(
getX() + getWidth(), getY(), 1, getHeight()) &&
moveSuccessfully(1,0)) {           setRunningAnimation();
} else {
    SpriteMidlet.vibrate(VIBRATION);
setStandingAnimation();
}
break;
default:
break;
}
}
private boolean moveSuccessfully(int dx, int dy) {
move(dx, dy);
if (canvas.overlapSpriteDog(this)) {
move(-dx, -dy);
return false;
} else {
return true;
}
}
private void setRunningAnimation() {
int[] sequence; if(currentDirection
== RIGHT) {       sequence =
animations[LEFT][RUN];
setTransform(TRANS_MIRROR);
} else {
    sequence = animations[currentDirection][RUN];
setTransform(TRANS_NONE);
}
 setFrame(sequence[(animationTick >> 1) % sequence.length]);
}
private void setStandingAnimation() {
if (currentDirection == RIGHT) {
setFrame(animations[LEFT][STAND][0]);
setTransform(TRANS_MIRROR);
} else {
    setFrame(animations[currentDirection][STAND][0]);
setTransform(TRANS_NONE);
}
}
}

```

---

**Listing VII-8** File SpriteGameCanvas.java

---

```
import javax.microedition.lcdui.Graphics; import
javax.microedition.lcdui.game.GameCanvas; import
javax.microedition.lcdui.game.LayerManager; import
javax.microedition.lcdui.game.Sprite;

public class SpriteGameCanvas extends GameCanvas
implements Runnable {

    private SpriteDog staticDog;
    private SpriteDog dynamicDog;
    private TiledField field;    private
LayerManager layerManager;

    public SpriteGameCanvas() {
super(true);      setFullScreenMode(true);
layerManager = new LayerManager();
field        = new TiledField();
staticDog    = new SpriteDog(this);
dynamicDog   = new SpriteDog(this);

    staticDog.setPosition(125,50);
    dynamicDog.setPosition(150,125);
    layerManager.append(staticDog);
    layerManager.append(dynamicDog);      layerManager.append(field);
    }
    protected void showNotify() {
new Thread(this).start();
    }
    public void run() {
        Graphics g = getGraphics();
        while(true) {
            int keyStates = getKeyStates() & ~FIRE_PRESSED;      int
direction = (keyStates == UP_PRESSED) ? SpriteDog.UP :
(keyStates == LEFT_PRESSED) ? SpriteDog.LEFT:
(keyStates == DOWN_PRESSED) ? SpriteDog.DOWN :
(keyStates == RIGHT_PRESSED) ? SpriteDog.RIGHT :
SpriteDog.NONE;      dynamicDog.tick(direction);

            field.tick();      render(g);
flushGraphics();      try {
Thread.sleep(25); }
catch(InterruptedException e){}
        }
    }
    public TiledField getField() {
return field;
    }
    public boolean overlapSpriteDog(Sprite sprite) {
return sprite.collidesWith(staticDog, false);
    }
    public void render(Graphics g) {
        g.setColor(255, 255, 255);
```

```
    g.fillRect(0, 0, getWidth(), getHeight());
layerManager.paint(g, 0, 0);
}
```

---

Kode program yang terakhir pada bab ini adalah `SpriteMidlet` yang digunakan untuk menjalankan aplikasi yang mendemokan penggunaan `Sprite`, *collision detection* dan efek spesial.

**Listing VII-9** File `SpriteMidlet.java`

```
import javax.microedition.lcdui.Command; import
javax.microedition.lcdui.CommandListener; import
javax.microedition.lcdui.Display; import
javax.microedition.lcdui.Displayable; import
javax.microedition.midlet.MIDlet;

public class SpriteMidlet extends MIDlet {

    private static Display display;

    protected void startApp() {
        SpriteGameCanvas canvas = new SpriteGameCanvas();
        canvas.addCommand(new Command("Exit", Command.EXIT, 0));

        canvas.setCommandListener(new CommandListener() {
            public void commandAction(Command c, Displayable d) {
                notifyDestroyed();
            }
        });
        display = Display.getDisplay(this);
        display.setCurrent(canvas);
    }

    public void destroyApp(boolean unconditional) {}
    public void pauseApp() {}

    public static void vibrate(int millis) {
        display.vibrate(millis);
    }
}
```



# VIII Deployment MIDlet

---

## VIII.1 Pengertian dan Cara Deployment MIDlet

*Deployment* MIDlet disini dapat disebut juga proses instalasi MIDlet pada perangkat *mobile*. Untuk melakukan instalasi ada beberapa cara yang dapat dilakukan. Semua cara ini belum tentu dapat dilakukan oleh suatu perangkat *mobile*. Beragamnya fitur dan kemampuan telepon genggam saat ini membuat ada jenis perangkat *mobile*, terutama yang keluaran lama, tidak mendukung beberapa *method* instalasi MIDlet. Cara-cara untuk instalasi MIDlet ke perangkat *mobile* antara lain sebagai berikut:

- a. Infrared
- b. Bluetooth
- c. Kabel data
- d. Over The Air provisioning (OTA)

Infrared, Bluetooth dan kabel data biasanya digunakan untuk *deployment* MIDlet dalam skala kecil artinya MIDlet diinstal ke satu perangkat pada satu waktu. OTA memungkinkan *deployment* secara masal. OTA menggunakan TCP/IP dalam pengoperasianya. MIDlet-MIDlet komersil biasanya menggunakan cara ini terutama terakhir. OTA dipilih karena merupakan cara yang mudah dan mampu menjangkau konsumen secara luas karena menggunakan jaringan internet. Lebih lanjut tentang OTA ini akan dibahas pada sub-bab over The Air Provisioning.

## VIII.2 File-File Deployment MIDlet

Sebelum suatu MIDlet/MIDlet *suite* di deploy kita harus menyiapkan file-filenya. Ada dua file yang harus dipersiapkan untuk deployment MIDlet yaitu:

- a. File JAR (Java Archive)
- b. File JAD (Java Application Descriptor)

File JAR adalah file yang berisi aplikasi MIDlet kita dan semua file-file yang dibutuhkan MIDlet kita untuk berjalan dengan baik seperti file-file gambar, audio atau video. Direktori/folder res yang menampung file-file pendukung ini dimasukkan ke dalam File JAR. Semua file .class hasil kompilasi kode MIDlet dimasukkan ke dalam file JAR ini.

File JAD berisi deskripsi tentang MIDlet yang akan diinstal. Keberadaan file ini bersifat opsional. Akan tetapi sangat dianjurkan setiap deployment MIDlet menyertakan file ini. Lebih lanjut mengenai file JAD ini dibahas di sub-bab berikutnya.

### VIII.3 File Java Application Descriptor (JAD)

File berisi keterangan yang mendeskripsikan MIDlet *suite* yang akan diinstal. Dalam satu MIDlet *suite* dapat terdapat beberapa MIDlet. Keterangan tentang MIDlet-MIDlet yang berada di dalam JAD ini antara lain nama MIDlet, ukuran, URL, pembuat, versi, format karakter yang digunakan dan lain-lain. Dalam satu file JAD dapat saja terdapat satu MIDlet.

Informasi yang terdapat pada File JAD didefinisikan dalam MIDP specification. Tetapi pada modul ini akan dibahas lima jenis yaitu informasi yang wajib ada, informasi mengenai MIDlet apa saja dalam MIDlet *suite* itu, informasi opsional, *informasi Over The Air* (OTA) dan informasi yang didefinisikan sendiri oleh pembuat MIDlet (*suite*).

Informasi yang wajib ada (*required information*) pada MIDlet *suite* antara lain:

- a. MIDlet-Name: Nama dari MIDlet *Suite*
- b. MIDlet-Version: Versi MIDlet *suite*/aplikasi
- c. MIDlet-Vendor: Vendor MIDlet *suite*
- d. MIDlet-Jar-URL: berisi tempat dimana file JAR dapat diambil
- e. MIDlet-Jar-Size: Ukuran dari MIDlet (JAR)
- f. Microedition-Profile: Versi MIDP yang digunakan
- g. Microedition-Configuration: Versi CLDC yang digunakan

Properti-properti opsional yang melekat pada MIDlet *suite* antara lain:

- a. MIDlet-Permissions: akses apa saja yang pasti akan diakses oleh MIDlet
- b. MIDlet-Permissions-Opt: akses tambahan apa saja yang mungkin diperlukan oleh MIDlet *suite*
- c. MIDlet-Description: deskripsi singkat tentang MIDlet *suite*
- d. MIDlet-Icon: icon yang akan dipakai oleh MIDlet *suite*
- e. MIDlet-Info-URL: URL tempat mendapatkan informasi tentang MIDlet *suite*

Informasi lain dalam JAD adalah untuk keperluan OTA. Properti-properti itu antara lain:

- a. MIDlet-Delete-Notify
- b. MIDlet-Delete-Confirm
- c. MIDlet-Instal-Notify

*Programmer/deployer* dapat mendefinisikan sendiri informasi yang diperlukan ke dalam JAD dengan format *key-value* sama seperti data pada Map dalam J2SE. Setelah file JAD berhasil di download dan

sebelum instalasi MIDlet *suite* akan dimunculkan layar konfirmasi pada perangkat mobile apakah ingin menginstal MIDlet *suite* tersebut.



**Gambar VIII-1** Konfirmasi yang menampilkan isi dari file JAD

**Listing VIII-1** Contoh File JAD

---

```
MIDlet-Name: Stock
MIDlet-Data-Size: 29 K
MIDlet-1: com.sun.java.Stock
MIDlet-Jar-URL: http://localhost:8080/midlets/stock.jar
MIDlet-Icon: /stock.png
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
MIDlet-Version: 2.0
MIDlet-Vendor: Sun Microsystems, Inc.
MIDlet-Description: View stocks prices over the internet
MIDlet-Info-URL: http://localhost:8080
MIDlet-Permissions: javax.microedition.io.Connector.http
```

---

## VIII.4 Keamanan Aplikasi MIDlet

MIDlet *suite* di perangkat *mobile* sebenarnya berjalan dalam *Sandbox* atau biasa disebut *Sandbox Environment*. Dalam *Sandbox* secara default MIDlet tidak diijinkan mengakses API-API yang dianggap sensitif. Pada spesifikasi MIDP 1.0, semua MIDlet dimasukkan dalam *Sandbox*. Pada spesifikasi MIDP 2.0, MIDlet dibagi menjadi dua jenis yaitu:

- a. *Untrusted MIDlet*: MIDlet yang tidak diijinkan mengakses API-API sensitif
- b. *Trusted MIDlet*: MIDlet yang diijinkan mengakses API-API sensitif

Pada MIDP 2.0 MIDlet dapat mengakses API-API yang sensitif dengan mengatur *permission* pada file JAD. *Permission* ini tidak diperlukan untuk mengakses API-API yang tidak sensitif. Di banyak perangkat meskipun sudah menggunakan *permission* pada file JAD-nya tetap saja pada saat pemanggilan fungsi seringkali memerlukan persetujuan eksplisit dari user.

Setiap MIDlet yang terinstal dalam suatu perangkat menempati suatu *protection domain*. *Protection domain* ini mempunyai serangkaian *permission* yang ditentukan berdasarkan *permission* yang dibutuhkan oleh masing-masing MIDlet dan mampu diberikan oleh perangkat tempat MIDlet terinstal.

*Permission* yang ada pada *protection domain* secara umum terbagi menjadi dua jenis yaitu:

- allowed: Ijin diberikan kepada MIDlet waktu mengakses API yang dia butuhkan tanpa persetujuan user/pengguna
- user: Ijin diberikan kepada MIDlet untuk mengakses API hanya dengan persetujuan user terlebih dahulu

*Method* pemberian ijin dari user/pengguna MIDlet terdiri dari tiga jenis yaitu:

- blanket: ijin diberikan cukup pertama kali dan selanjutnya ijin akan otomatis diberikan ke MIDlet tanpa keterlibatan user lagi sampai MIDlet di uninstal
- session: ijin diberikan untuk setiap kali pemanggilan MIDlet
- oneshot: ijin diberikan setiap kali MIDlet akan mengakses fungsi API yang sensitif

**Tabel VIII-1** API yang tidak membutuhkan *permission* dalam penggunaannya

API	Deskripsi
javax.microedition.lcdui	Interface pengguna baik High-level maupun Low-level
javax.microedition.lcdui.game	
javax.microedition.media	Multimedia kontrol baik itu untuk audio maupun video
javax.microedition.media.control	
javax.microedition.rms	API Record Manajemen System
javax.microedition.midlet	API dasar MIDlet

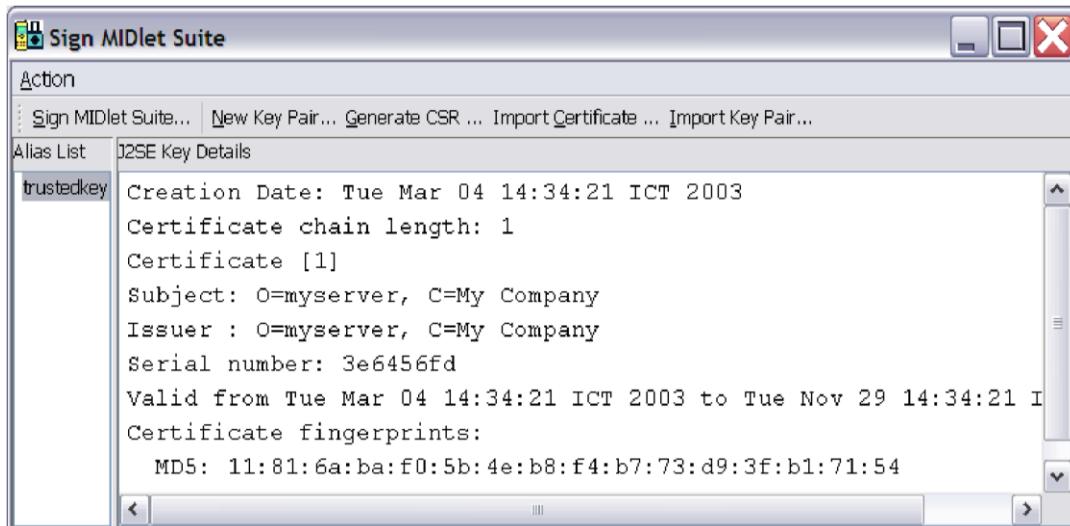
**Tabel VIII-2** API yang membutuhkan *permission* dalam penggunaannya

API	Permission
javax.microedition.io.SocketConnection	javax.microedition.io.Connector.socket
javax.microedition.io.ServerSocketConnection	javax.microedition.io.Connector.serversocket

javax.microedition.io. SecureConnection	javax.microedition.io.Connector.ssl
javax.microedition.io. CommConnection	javax.microedition.io.Connector. comm.
javax.microedition.io. HttpConnection	javax.microedition.io.Connector. http
javax.microedition.io. HttpsConnection	javax.microedition.io.Connector. https
javax.microedition.io. UDPDatagramConnection	javax.microedition.io.Connector. datagram
javax.microedition.io. UDPDatagramConnection	javax.microedition.io.Connector. Datagramreceiver
javax.microedition.io.PushRegistry	javax.microedition.io.PushRegistry

Otentikasi MIDlet pada perangkat mobile dapat menggunakan mekanisme otentikasi yang sama pada web yaitu dengan X.509 PKI. Pada MIDP 2.0 diperkenalkan fitur yang memungkinkan MIDlet memanfaatkan kelebihan dari digital signature. Dengan penggunaan *digital signature* dapat dilakukan otentikasi MIDlet sebelum MIDlet itu diinstal pada perangkat bergerak. Fasilitas *digital signature* ini disediakan pada WTK. Key untuk *digital signature* dapat didapatkan menggunakan *key generator* pada

WTK atau dengan memesannya melalui vendor *security* seperti Verisign atau Thawte.



**Gambar VIII-2** *Digital Signature* pada WTK



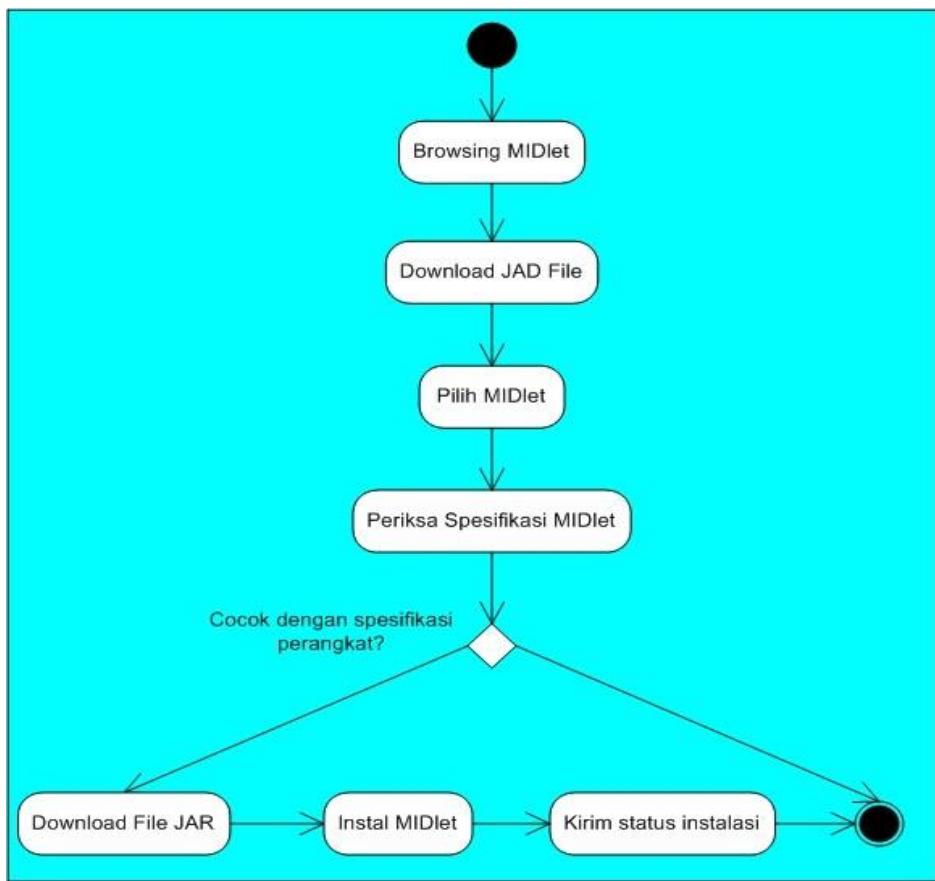
Gambar VIII-3 Pembentukan Key-Pair untuk Digital Signature pada WTK

### VIII.5 Over The Air Provisioning

*Over The Air provisioning* adalah proses *deployment* MIDlet (*suite*) dengan menggunakan jaringan operator seluler. OTA ini adalah *method deployment* yang paling sering digunakan untuk men-deploy MIDlet. *Method* ini yang paling sering digunakan oleh *content provider* untuk men-deploy MIDlet mereka. *Method* ini dipilih karena mampu menjangkau konsumen secara luas.

Pada MIDP 1.0 OTA belum masuk spesifikasi hanya menjadi tambahan saja. Pada MIDP 2.0 OTA masuk dalam spesifikasi untuk memberi keseragaman dalam *deployment* MIDlet. Pada spesifikasi MIDP 2.0 OTA melalui protokol HTTP dalam pelaksanaannya biasa digunakan WWW, WAP ataupun iMode browser. Tahap-tahap instalasi melalui OTA biasanya dilakukan sebagai berikut:

1. Browsing MIDlet Download JAD
2. Periksa spesifikasi MIDlet berdasarkan JAD
3. Kalau spesifikasi JAD cocok, download file JAR MIDlet
4. Pengiriman status instalasi ke OTA server



**Gambar VIII-4** Proses instalasi via OTA

**Tabel VIII-3** Status instalasi melalui proses OTA

Kode Status	Deskripsi
<b>900</b>	Instalasi Sukses
<b>901</b>	Memori tidak mencukupi
<b>902</b>	Pembatalan oleh pengguna/user
<b>903</b>	<i>Loss of Service</i>
<b>904</b>	Ukuran File JAR tidak cocok
<b>905</b>	Atribut tidak cocok
<b>906</b>	File Descriptor tidak valid
<b>907</b>	File JAR tidak valid
<b>908</b>	Configuration atau Profile dari perangkat tidak sesuai
<b>910</b>	Otentikasi Aplikasi MIDlet mengalami kegagalan
<b>911</b>	Kegagalan Push Registration

**922**

Notifikasi penghapusan/uninstall



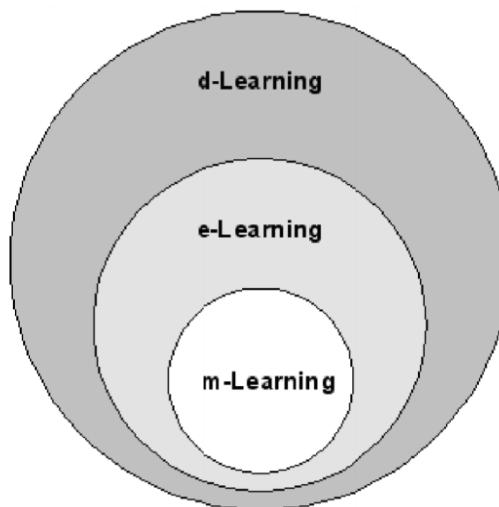
# IX Pengenalan Mobile Learning

---

## IX.1 Pengertian *m-Learning*

Perkembangan teknologi informasi dan komunikasi telah memberi pengaruh besar bagi dunia pendidikan dan pelatihan. Munculnya teknologi telah melahirkan *Computer Based Learning* (CBL), pembelajaran yang didukung oleh perangkat dan media elektronik digital, *e-Learning* (yang terutama memanfaatkan internet dan LMS) sampai *m-Learning* yang merupakan pembelajaran yang didukung oleh perangkat bergerak dan transmisi nirkabel. Perangkat bergerak telah muncul sebagai salah satu teknologi yang cukup potensial untuk mendukung pembelajaran.

Istilah *mobile learning* (*m-Learning*) mengacu kepada penggunaan perangkat teknologi informasi (TI) genggam dan bergerak, seperti PDA, telepon genggam, laptop dan tablet PC, dalam pengajaran dan pembelajaran. *m-Learning* merupakan bagian dari *e-Learning* sehingga, dengan sendirinya, juga merupakan bagian dari *distant learning* (*d-learning*) (Gambar 1).



**Gambar IX-1** Set/himpunan *d-Learning*

Beberapa kemampuan penting yang harus disediakan oleh perangkat pembelajaran *m-Learning* adalah adanya kemampuan untuk terkoneksi ke peralatan lain (terutama komputer), kemampuan menyajikan informasi pembelajaran dan kemampuan untuk merealisasikan komunikasi bilateral antara pengajar dan pembelajar.

## **IX.2 Kelebihan dan Kekurangan M-Learning**

Dalam pembelajaran *e-Learning*, independensi waktu dan tempat menjadi faktor penting yang sering ditekankan. Namun, dalam *e-Learning* tradisional kebutuhan minimum tetap sebuah PC yang memiliki konsekuensi bahwa independensi waktu dan tempat tidak sepenuhnya terpenuhi. Independensi ini masih belum dapat dipenuhi dengan penggunaan *notebook* (komputer portabel), karena independensi waktu dan tempat yang sesungguhnya berarti seseorang dapat belajar dimana-pun kapan-pun dia membutuhkan akses pada materi pembelajaran. Tiga kalangan diidentifikasi sebagai pengguna potensial *m-Learning* yaitu, pelajar/akademisi, pekerja dan pembelajar sepanjang hayat (*lifelong learner*).

*M-Learning* adalah pembelajaran yang unik karena materi pembelajaran, arahan dan aplikasi yang berkaitan dengan pembelajaran, kapan-pun dan dimana-pun dapat diakses. Hal ini akan meningkatkan perhatian pada materi pembelajaran, membuat pembelajaran menjadi pervasif, dan dapat mendorong motivasi pembelajar kepada pembelajaran sepanjang hayat (*lifelong learning*). Selain itu, dibandingkan pembelajaran konvensional, *m-Learning* memungkinkan adanya lebih banyak kesempatan untuk kolaborasi secara *ad hoc* dan berinteraksi secara informal di antara pembelajar.

Beberapa kelebihan *m-Learning* dibandingkan dengan pembelajaran lain adalah:

- a. Dapat digunakan dimana-pun pada waktu kapan-pun,
- b. Kebanyakan perangkat bergerak memiliki harga yang relatif lebih murah dibanding harga PC *desktop*,
- c. Ukuran perangkat yang kecil dan ringan daripada PC *desktop*,
- d. Diperkirakan dapat mengikutsertakan lebih banyak pembelajar karena *m-Learning* memanfaatkan teknologi modern yang biasa digunakan dalam kehidupan sehari-hari.

Meski memiliki beberapa kelebihan, *m-Learning* tidak akan sepenuhnya menggantikan *e-Learning* tradisional. Hal ini dikarenakan *m-Learning* memiliki keterbatasan-keterbatasan terutama dari sisi perangkat/media belajarnya. Keterbatasan perangkat bergerak antara lain sebagai berikut.

- a. Kemampuan prosesor
- b. Kapasitas memori
- c. Layar tampilan
- d. Kemampuan sumber daya listrik
- e. Perangkat I/O

*M-Learning* akan cukup tepat jika diterapkan di lingkungan di mana *computer aided learning* tidak tersedia. Hal ini dikarenakan pengguna yang telah terbiasa dengan penggunaan PC sebagai media belajarnya, ternyata lebih suka tetap memakai PC, sedangkan mereka yang tidak familiar dengan PC merasa penggunaan perangkat bergerak lebih atraktif dan lebih dapat diterima. Sistem yang optimal adalah

menggabungkan *m-Learning* dengan *e-Learning*, di mana ada alternatif proses pembelajaran dilakukan dengan perangkat komputer dan/atau perangkat bergerak atau digabungkan dengan sistem tradisional.

### IX.3 Klasifikasi M-Learning

*M-Learning* dapat dikelompokkan dalam beberapa klasifikasi tergantung dari beberapa sudut pandang. Dari sisi teknologi ICT yang digunakan, maka *m-Learning* dapat diklasifikasi berdasar indikator utama;

tipe perangkat yang didukung - *laptop*, *tabletPC*, *PDA (Personal Digital Assistant)*, telepon seluler atau *smartphone*; tipe komunikasi *wireless* yang digunakan untuk mengakses materi pembelajaran dan informasi adminitrasi, yaitu *Global System for Mobile (GSM)*, *GPRS*, *Universal Mobile Telecommunication System (UMTS)*, *Code Division Multiple Access (CDMA)*, *Institute of Electrical and Electronics Engineers (IEEE) 802.11*, *Bluetooth*, *Infrared Data Access (IrDA)* dan sebagainya.

Sedangkan dari sisi pandang teknologi pendidikan klasifikasi berdasarkan indikator utama seperti tersebut di bawah ini :

- a. Waktu komunikasi antara pengajar dan pembelajar,
- b. Standar *e-Learning* yang mendukung,
- c. Sifat pengaksesan,
- d. Lokasi pengguna yang dapat menggunakan sistem *m-Learning*,
- e. Informasi yang dapat diakses.

Klasifikasi berdasarkan waktu komunikasi antara pengajar dan pembelajar sistem *m-Learning* diklasifikasikan sebagai berikut.

- a. Sistem yang mendukung pendidikan *synchronous*, yaitu yang mampu menyediakan komunikasi antara sesama pelajar dan pengajar dengan pelajar secara *realtime*. Komunikasi tersebut dapat berupa komunikasi suara, video maupun *chat*.
- b. Sistem yang mendukung pendidikan *asynchronous*, dimana komunikasi antara sesama pelajar atau pelajar dengan pengajar bersifat tidak *realtime*. Komunikasinya biasanya melalui *Short Message Service (SMS)* atau email yang pengiriman informasinya secara *asynchronous*.
- c. Sistem yang mendukung pendidikan *asynchronous* dan *synchronous*.

Pada saat ini belum ada spesifikasi dan standar *m-Learning*, maka klasifikasi sistem *m-Learning* dapat berdasarkan dukungan sistem terhadap standar *e-Learning*, yaitu :

- a. Sistem *m-Learning* yang tidak mendukung spesifikasi dan standar *e-Learning* (*Shareable Courseware Object Reference Model (SCORM)*, *Aviation Industry CBT Committee (AICC)* dan lain sebagainya);
- b. Sistem *m-Learning* yang mendukung spesifikasi dan standar *e-Learning*.

Berdasarkan sifat pengaksesan materi dan informasi, sistem *m-Learning* dapat diklasifikasikan sebagai berikut.

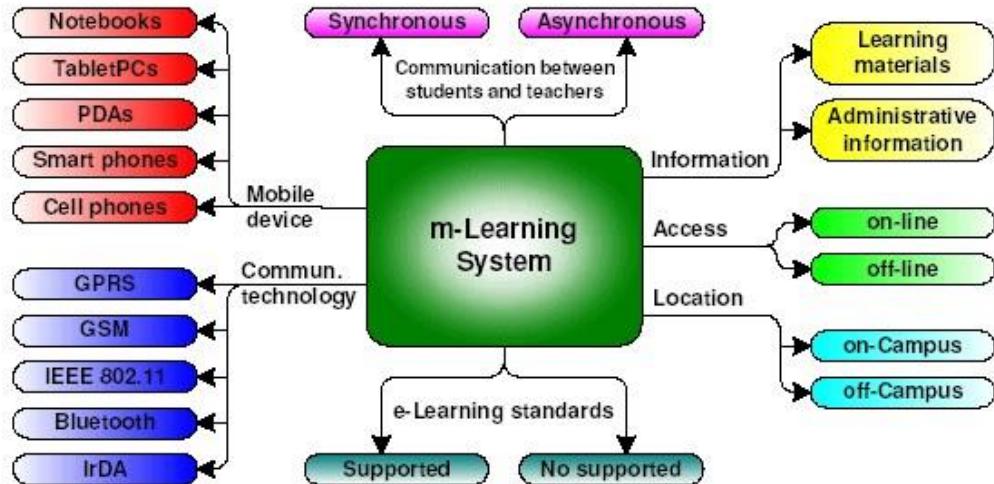
- a. Sistem *m-Learning online*. Sistem ini membutuhkan koneksi yang permanen antara sistem dan perangkat pengguna.
- b. Sistem *m-Learning offline*. Pada sistem ini, materi dapat di-upload dari perangkat *mobile* tanpa harus berkomunikasi dengan sistem dengan *wireless*.
- c. Sistem *m-Learning* dengan *offline* dan *online*. Untuk mengakses suatu bagian materi pembelajaran dengan cara *online* sedangkan untuk mengakses kembali materi tersebut adalah dengan cara *offline*.

Berdasarkan lokasi dari pengguna, sistem *m-Learning* dapat diklasifikasikan menjadi tiga kelompok, yaitu:

- a. Dalam kampus, yang hanya dapat dari dalam kampus atau sekolah. Tipe cara aksesnya dengan menggunakan *laptop* atau tablet PC dan melalui jaringan *wireless* kampus;
- b. Luar kampus, yang dapat diakses dari luar kampus atau sekolah. Cara aksesnya dengan menggunakan PDA, telepon seluler atau *smartphone* yang mendukung komunikasi *wireless* jarak jauh;
- c. Sistem yang dapat diakses dari dalam dan luar kampus atau sekolah.

Berdasarkan informasi yang diakses, sistem *m-Learning* dapat diklasifikasikan menjadi tiga kelompok, yaitu :

- a. Sistem *m-Learning* yang mendukung pengaksesan konten pendidikan seperti materi, ujian, kamus dan sebagainya.
- b. Sistem *m-Learning* yang mendukung pengaksesan terhadap pelayanan administrasi seperti jadwal, nilai dan lain sebagainya.
- c. Sistem *m-Learning* yang dapat mendukung pengaksesan terhadap konten pendidikan dan pelayanan administrasi.



Gambar IX-2 Klasifikasi *m-Learning*

#### IX.4 Metode Penyusunan M-Learning

Terdapat beberapa metoda untuk menyusun dan mengorganisasi sebuah aplikasi pembelajaran berbantuan komputer dan berbasis multimedia (*computer-aided and multimedia based application*), termasuk aplikasi *m-Learning*. Meisenberger mengkategorikan 4 (empat) metoda dalam penyusunan dan pengorganisasian aplikasi *m-Learning*. Metoda tersebut adalah sebagai berikut:

- Eksposisi
- Eksplorasi
- Konstruksi
- Komunikasi

Metoda eksposisi mengorganisasi obyek pembelajaran (*learning object*) dalam suatu jalur pembelajaran (*learning path*) yang berurutan (*sequential*) dan terdefinisi. Untuk menyajikan sebuah obyek pembelajaran di lingkungan *m-Learning* dapat menggunakan cara yang sama dengan *e-Learning*. Perbedaan utama dengan *e-Learning* terletak pada layar tampilan yang terbatas sehingga obyek pembelajaran dalam *m-Learning* seharusnya menggunakan teks yang singkat dan tidak terlalu panjang serta disertai dengan gambar-gambar yang ilustratif. Topik yang besar hendaknya dipecah ke dalam topik-topik yang lebih kecil dan independen. Dengan asumsi bahwa pengguna *m-Learning* akan lebih banyak menggunakan periode *idle* (*idle periods*) yang mana waktu dan konsentrasi belajar sangat terbatas, topik belajar yang kecil dan independen akan lebih mudah diterima. Belajar dengan eksposisi melalui sebuah obyek pembelajaran lebih sesuai untuk diterapkan bagi pembelajar yang belum berpengalaman.

Metoda eksplorasi atau belajar secara eksploratif dapat diartikan sebagai belajar yang tanpa menggunakan jalur pembelajaran yang runtut dan membebaskan pembelajar untuk mengeksplorasi konten pembelajaran sesuai keinginan dan kebutuhannya. Realisasinya dapat berupa sebuah sistem yang

terkoneksi dengan jaringan atau sebuah *knowledge system* dengan disertai fasilitas pencarian interaktif. Belajar secara eksploratif cocok untuk diterapkan bagi pembelajaran yang telah memiliki pengetahuan dan keterampilan untuk belajar secara mandiri. Contoh sederhana dari kategori ini adalah kamus istilah (*glossary*), baik yang terkoneksi ke *database* atau sumber daya lain, seperti internet, yang memungkinkan pembelajar untuk melakukan pencarian definisi suatu istilah secara interaktif. Penerapan metoda ini di lingkungan *m-Learning* memiliki permasalahan yang sama dengan kategori eksposisi yaitu waktu dan konsentrasi belajar sangat terbatas.

Metode konstruksi atau belajar secara konstruktif dapat digambarkan sebagai "belajar dengan bertindak" atau "*learning by doing*". Instrumen utama belajar secara konstruktif adalah pemodelan dan simulasi. Permasalahan utamanya, untuk saat sekarang ini, perangkat pembelajaran *m-Learning* masih belum memiliki *processing power* yang cukup memungkinkan untuk mengimplementasikan hal tersebut.

Metode komunikasi, seperti namanya, lebih menitik beratkan pada aspek komunikasi. Komunikasi merupakan aspek penting dari sebuah pembelajaran. Perangkat komunikasi di lingkungan *m-Learning* memungkinkan adanya diskusi dan berbagi pengetahuan baik antara pembelajar dengan pengajar maupun antar sesama pembelajar sendiri dengan lebih mudah, di mana-pun dan kapan-pun. Komunikasi di lingkungan *mobile* bukan menjadi masalah besar mengingat perangkat *mobile* memang dirancang sebagai perangkat komunikasi. Komunikasi sendiri dapat dilakukan secara *synchronous* maupun *asynchronous*.

## IX.5 Aspek Perancangan M-Learning

Pengembangan aplikasi *m-Learning* dan aplikasi perangkat bergerak secara umum sangat berbeda dengan pengembangan aplikasi untuk PC *desktop*. Aplikasi pada perangkat bergerak memiliki keterbatasan-keterbatasan yang memaksa pengembang untuk menyesuaikan diri dengan karakteristik yang dimilikinya. Suatu konten pembelajaran *e-Learning* tidak dapat dengan serta merta dipakai dalam *m-Learning* namun harus diadaptasikan terlebih dahulu. Beberapa aspek yang menjadi pertimbangan pada saat perancangan *m-Learning* adalah sebagai berikut:

- a. Keterbatasan hardware
- b. Keterbatasan jaringan
- c. Perangkat yang pervasif
- d. Skema integrasi
- e. Kenyamanan pengguna

## IX.6 Keterbatasan Hardware

Perbedaan yang paling mencolok antara *platform* bergerak dan komputer *desktop* adalah pada *computing resources* yang dimiliki. perangkat bergerak memiliki kemampuan pemrosesan dan sumber daya yang

terbatas. Keterbatasan ini dapat dilihat pada pemroses, kapasitas penyimpanan, perangkat *input/output*. Pengembang harus mempertimbangkan beberapa aspek terkait dengan keterbatasan *hardware* ini. Aspek tersebut antara lain adalah sebagai berikut.

Penggunaan pustaka yang tepat. Pengembangan aplikasi seringkali membutuhkan pustaka *software* yang dikembangkan oleh pihak ketiga. Saat ini terdapat beragam pustaka yang dapat digunakan yang dapat dipilih sesuai kebutuhan. Pemilihan pustaka yang tepat untuk mendukung fitur yang dibutuhkan aplikasi dengan kebutuhan *hardware* yang paling minimal merupakan faktor paling esensial. Kekurangtepatan pemilihan pustaka akan dapat menyebabkan aplikasi menjadi besar dan lambat bahkan sulit untuk *deploy* ke dalam perangkat.

Pengurangan *footprint* aplikasi sangat penting dilakukan. Keterbatasan penyimpanan pada perangkat bergerak mengharuskan minimisasi penggunaan penyimpanan (*storage*) maupun saat *runtime*. Untuk mengatasi hal ini, solusi yang dapat dilakukan adalah : *pertama*, dengan melakukan optimisasi proses pemaketan (*packaging*), yaitu memastikan hanya bagian tertentu dari pustaka yang benar-benar dibutuhkan yang disertakan dalam aplikasi. *Kedua*, dengan memecah aplikasi ke dalam beberapa bagian terpisah yang memungkinkan hanya aplikasi yang diperlukan saja yang di-*load* pada saat *runtime*.

Pemanfaatan portal cukup membantu dalam m-Learning. Meski perkembangan teknologi perangkat semakin canggih, bagaimanapun, untuk proses-proses tertentu yang bersifat kompleks akan masih memerlukan sumber daya yang tidak mampu diatasi oleh perangkat. Pendekatan yang umum digunakan untuk masalah ini adalah penggunaan *server* portal untuk mendelegasikan tugas-tugas yang kompleks.

## IX.7 Keterbatasan Jaringan

Berbeda dengan jaringan *broadband* yang biasa digunakan pada jaringan komputer *desktop* dan *server*, jaringan seluler memiliki kekurangan di antaranya adalah sangat lambat, tak dapat diandalkan dan tidak aman. Keadaan ini menyebabkan pengembangan aplikasi pada perangkat bergerak harus memanfaatkan sumber daya jaringan secara maksimal. Beberapa aspek pengembangan aplikasi bergerak yang harus diperhatikan terkait pada keterbatasan jaringan adalah sebagai berikut.

- a. Dukungan pada mode *offline*. Koneksi jaringan yang tidak stabil mengharuskan aplikasi memiliki kapabilitas untuk mendukung operasi *offline* pada saat jaringan tidak tersedia. Teknologi *enabler* yang memungkinkan hal ini adalah kemampuan penyimpanan *on-device*, yakni data aplikasi disimpan secara lokal pada devais dan hanya memerlukan waktu-waktu tertentu untuk terkoneksi ke jaringan (*occasionally connected*). Selain untuk mengatasi ketidakstabilan jaringan, penyimpanan *on-device* juga dapat mengurangi lalu-lintas jaringan serta meningkatkan unjuk kerja.
- b. I/O tersangga (*buffered*). Pembacaan data secara per-*byte* memerlukan waktu relatif lama.

Keadaan ini memerlukan penulisan dan pembacaan data dalam potongan-potongan (*chunk*).

- c. Enkripsi data. Penggunaan jaringan nirkabel memiliki resiko keamanan lebih besar karena siapapun memiliki kemungkinan untuk menyadap data sehingga perlu adanya mekanisme keamanan untuk melindungi data. Namun, keamanan akan mempengaruhi unjuk kerja. Pada devais yang kecil perlu kehati-hatian dalam mengevaluasi kebutuhan kemanan serta pemilihan solusi yang sepadan.
- d. Mendapatkan status *server* dengan efisien. Pada beberapa aplikasi tertentu kadang membutuhkan adanya *update* dengan status *server* secara *real time*. Frekuensi *polling* diharapkan lebih cepat dari perkiraan perubahan status *server*. *Polling* secara terus menerus akan banyak memakan banyak *bandwidth*, sumberdaya dan waktu. Hal ini membutuhkan mekanisme *polling* yang efisien.

## IX.8 Perangkat yang Pervasif

Tidak seperti komputer *desktop* yang dapat diadministrasi secara terpusat, pengelolaan perangkat yang kecil yang selalu dibawa ke mana-mana oleh penggunanya merupakan persoalan yang lebih rumit. Persoalan dalam pengelolaan perangkat ini dapat muncul baik persoalan sosial maupun persoalan teknis. Beberapa aspek yang perlu diperhatikan adalah sebagai berikut.

- a. Proteksi data *on-device*. Perangkat yang kecil sangat mudah untuk hilang. Data *on-device* dapat berupa data sensitif yang dapat menimbulkan resiko kemanan yang besar. Untuk melindungi data *on-device* dari kemungkinan diakses oleh pihak-pihak yang tidak memiliki otoritas, maka perlu adanya mekanisme enkripsi pada data *on-device* tersebut.
- b. Optimasi pada banyak perangkat. Beragamnya perangkat mengharuskan aplikasi yang dibuat dapat dijalankan pada perangkat yang berbeda-beda. Aplikasi yang dibangun hendaknya memiliki *platform* yang *device-independent*.

## IX.9 Skema Integrasi

Banyak aplikasi nirkabel bergerak yang membutuhkan integrasi dengan banyak sistem *back-end* atau *middleware* berbeda. Terdapat beberapa teknologi integrasi yang dapat digunakan, antara lain adalah sebagai berikut.

- a. Protokol biner *proprietary*. Dukungan HTTP yang dimiliki oleh sebagian besar perangkat adalah basis bagi pendekatan kebanyakan protokol. Dengan menggunakan *transport* HTTP aplikasi dapat dibuat dengan merancang protokol sesuai kebutuhan dan meminimalisasi jumlah *byte* yang harus dikirimkan pada jaringan. Namun, pendekatan ini memerlukan perancangan dengan *coupling* yang ketat (*tightly coupled*) di mana pengembang harus mengembangkan komponen aplikasi baik pada sisi *server* maupun klien secara bersama-sama dengan antarmuka dengan protokol yang sesuai. Pengubahan disain aplikasi memerlukan pengubahan aplikasi pada kedua sisi.

- b. *Framework RPC*. RPC merupakan pendekatan integrasi yang lebih standar dan telah tersedia secara komersil. RPC mendefinisikan kumpulan API yang memungkinkan *server* dan klien melakukan pemanggilan dan melewatkkan parameter RPC. Penggunaan *framework* RPC dapat menghemat waktu pengembangan komponen antarmuka yang *proprietary* dan sulit dikelola. Akan tetapi, antara *server* dan klien masih bersifat *tightly coupled*.
- c. *Messaging*. Solusi *messaging* memisahkan (*decouple*) antara klien dan *server* melalui *messaging middleware*. Disain *messaging* dapat meningkatkan reliabilitas dan skalabilitas sistem karena sumber daya dapat dialokasikan untuk merespon sebuah permintaan (*request*) pada basis prioritas daripada basis *first-time-first-served*.
- d. XML dan *web services*. XML *web services* mendukung integrasi gaya RPC maupun *messaging* sekaligus. Namun, XML *web services* membutuhkan bandwith besar serta *overhead* CPU. Saat ini terdapat beberapa protokol dan implementasi XML *webservices* yang dapat digunakan pada aplikasi bergerak.

**Tabel IX-1** Perbandingan teknologi integrasi

Skema	Interoperabilitas	Kopling	Footprint
<b><i>Binary over HTTP</i></b>	Kurang	Ketat	Ringan
<b><i>RPC Framework</i></b>	Bagus	Ketat	Ringan
<b><i>Messaging</i></b>	Bagus	Longgar	Bagus
<b><i>XML Web Services</i></b>	Sangat bagus	Longgar	Berat

## IX.10 Kenyamanan Pengguna

Kenyamanan penggunaan aplikasi yang dapat dipakai “kapan-pun, dimana-pun” merupakan kekuatan terbesar dari aplikasi bergerak. Merancang aplikasi yang nyaman digunakan merupakan tantangan besar bagi para pengembang. Beberapa isu terkait dengan hal ini adalah sebagai berikut.

- a. Antarmuka pengguna yang kaya
- b. Pemanfaatan *thread*
- c. Satu layar setiap saat
- d. Menyimpan preferensi pengguna
- e. Menggunakan *deployment descriptor*

Karakteristik yang dimiliki oleh perangkat mengharuskan sistem maupun konten pembelajaran harus didisain secara khusus sesuai kemampuan dan keterbatasan yang dimiliki oleh perangkat.





# X Konten Mobile Learning

---

## X.1 Sekilas tentang Konten m-learning

Suatu m-Learning bagaimanapun canggihnya teknologi yang digunakan akan menjadi sia-sia tanpa didukung konten yang bagus. Konten pembelajaran dalam m-Learning memiliki jenis bermacam-macam. Konten sangat terkait dengan kemampuan perangkat untuk menampilkan atau menjalankannya. Keragaman jenis konten ini mengharuskan pengembang untuk membuat konten-konten yang tepat dan sesuai dengan karakteristik perangkat maupun pengguna. Jenis-jenis konten *m-Learning* antara lain: a. Teks

- b. Gambar
- c. Audio
- d. Video
- e. Aplikasi

## X.2 Konten Teks

Kebanyakan perangkat saat ini telah mendukung penggunaan teks. Hampir semua telepon seluler yang beredar saat ini telah mendukung penggunaan SMS. Kebutuhan memori yang relatif kecil memuat konten berbasis teks lebih mudah diimplementasikan. Namun keterbatasan jumlah karakter yang dapat ditampilkan harus menjadi pertimbangan dalam menampilkan konten pembelajaran sehingga perlu strategi khusus agar konten pembelajaran dapat disampaikan secara tepat dan efektif meskipun dengan keterbatasan ini. Salah satu contoh aplikasi pembelajaran berbasis teks/SMS adalah StudyTXT yang dikembangkan di salah satu Universitas di Selandia baru.



**Gambar X-1** Konten berbasis teks

### X.3 Konten Gambar

Perangkat bergerak yang ada sekarang telah banyak mendukung pemakaian gambar. Kualitas gambar yang dapat ditampilkan dapat beragam dari tipe monokrom sampai gambar berwarna berkualitas tinggi tergantung kemampuan perangkat. File gambar yang didukung oleh perangkat umumnya bertipe PNG, GIF, JPG. Penggunaan gambar sebagai konten pembelajaran biasanya digabungkan dengan konten lain, misalnya teks.



**Gambar X-2** Perangkat bergerak menampilkan gambar

### X.4 Konten Audio Video

Banyak perangkat bergerak saat ini telah mendukung penggunaan audio. beberapa tipe file yang biasanya digunakan di lingkungan perangkat bergerak antara lain rm, mp3, amr dan lain-lain. Oleh karena file audio biasanya memiliki ukuran yang cukup besar hal ini menyebabkan file audio tersebut harus diolah terlebih dahulu sehingga dapat digunakan di lingkungan perangkat bergerak yang memiliki kapasitas memori yang relatif kecil.

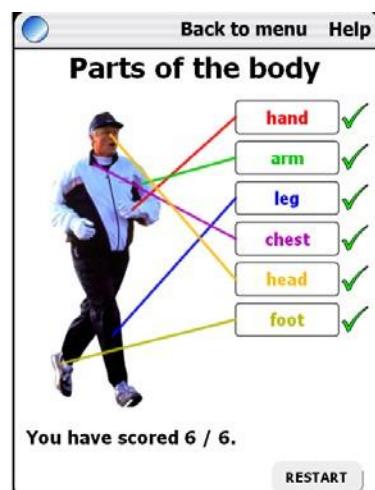
Meski dalam kualitas dan ukuran yang teratas, beberapa tipe perangkat bergerak telah mampu memainkan file video. Format file yang didukung oleh perangkat bergerak antara lain adalah 3gp, MPEG, MP4, dan lain-lain. Sama seperti file audio, kebanyakan file video memiliki ukuran yang cukup besar sehingga harus dikonversi dan disesuaikan dengan keterbatasan perangkat.



**Gambar X-3** Perangkat bergerak menampilkan video

## X.5 Konten Aplikasi

Konten yang cukup menarik adalah aplikasi perangkat lunak yang dipasang pada perangkat. Perangkat lunak dapat dikostumisasi sesuai kebutuhan sehingga akan lebih mudah dan intuitif untuk digunakan. Aplikasi perangkat lunak ini juga mampu menggabungkan konten-konten lain seperti teks, audio dan video sehingga menjadi lebih interaktif. Jenis aplikasi yang saat ini banyak digunakan antara lain aplikasi berbasis WAP/WML, aplikasi Java, aplikasi Symbian, dan lain-lain.



**Gambar X-4** Aplikasi perangkat lunak dijalankan pada perangkat

## X.6 Perangkat Pengembangan Konten

Untuk mengembangkan konten diperlukan perangkat pengembangannya. Perangkat pengembangan ini terdiri dari beberapa jenis perangkat yaitu editor, konverter dan simulator. Editor dan konverter biasanya

merupakan satu perangkat sedangkan simulator adalah perangkat terpisah. Simulator merupakan sarana testing pertama untuk konten yang telah kita kembangkan. Seringkali setelah proses simulasi konten perlu dikembangkan lebih lanjut.

Editor dan konverter dibedakan berdasarkan konten yang dikembangkannya. Ada tiga jenis editor konverter berdasarkan konten yaitu editor dan konverter untuk konten gambar, audio dan video. Untuk konten yang berupa aplikasi maka editor dan konverternya merupakan perangkat pengembangan perangkat lunak (*software development kit*) sesuai dengan *platform* tempat konten tersebut akan diimplementasikan.

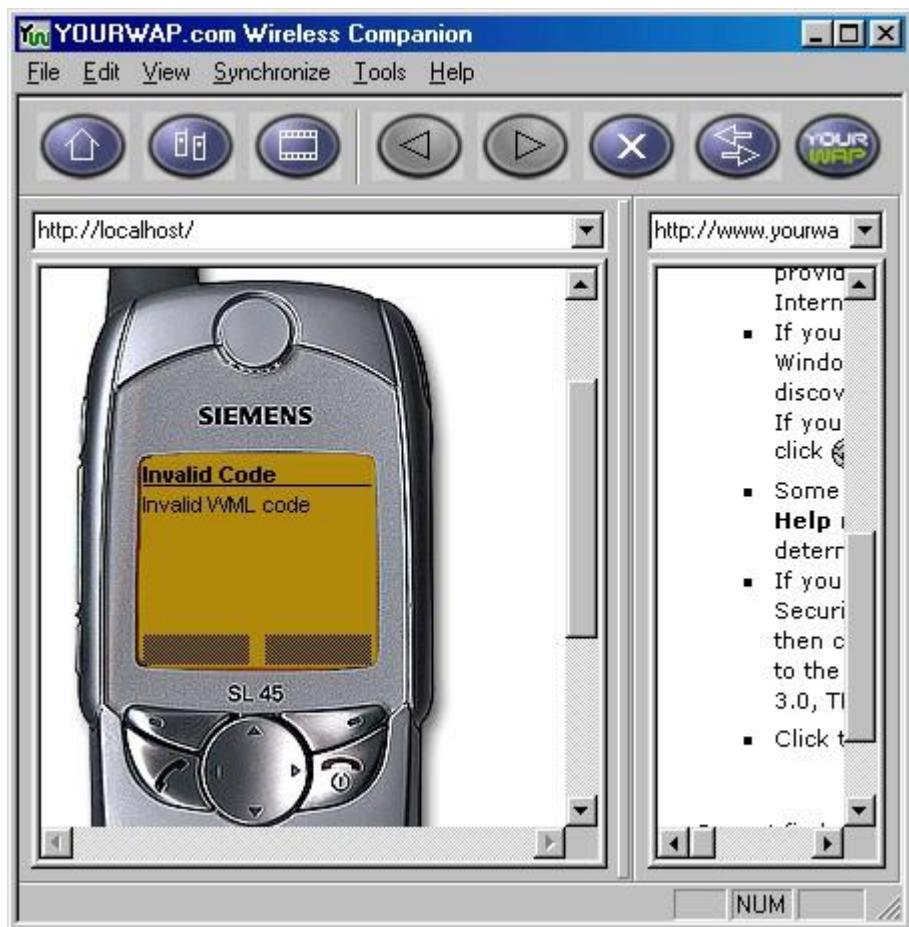
Editor dan konverter untuk konten yang berupa gambar memerlukan bantuan perangkat lunak untuk rekayasa grafis. Untuk membuat sebuah gambar yang sesuai dengan kemampuan perangkat diperlukan perangkat bantu yang memiliki fitur-fitur terutama untuk kostumisasi ukuran dan kualitas gambar serta untuk menyimpan gambar tersebut ke dalam format file yang didukung perangkat. Contoh perangkat lunak yang dapat digunakan untuk keperluan ini adalah Adobe Photosop, ImageForge, dan lain-lain.

Editor dan Konverter Audio merupakan perangkat lunak bantu yang harus mampu melakukan pengolahan file-file audio. File audio dapat diolah dan dikonversi ke dalam berbagai format. File audio ini dapat diolah dengan memotong, menggabung, mengubah kualitas, dan menyimpan ke format yang didukung oleh perangkat. Contoh perangkat lunak untuk mengedit dan mengkonversi audio adalah AudaCity, MP3CutterJoiner, Wavepad, Nokia Multimedia Converter, dan lain-lain.

Editor dan Konverter Video berfungsi untuk mengedit dan mengkonversi file video dapat digunakan beberapa perangkat lunak di antaranya adalah Nokia Multimedia Converter, MediCoder, ImToo, dan lain-lain.

Salah satu contoh simulator konten adalah WAP Simulator. Konten berbasis WAP merupakan konten online yang disimpan pada sebuah server web dan ditampilkan di perangkat menggunakan WAP browser pada perangkat yang terkoneksi ke server menggunakan teknologi koneksi tertentu, misalnya GPRS, CSD, 3G dan lain-lain. Konten berbasis WAP dibuat dengan spesifikasi khusus menggunakan format WML. Sebuah file WML dapat dibuat menggunakan editor teks biasa, semisal Notepad. File ini kemudian disimpan ke dalam ekstensi WML. Untuk menampilkan konten dari file dengan format WML ini, file tersebut harus di-upload terlebih dahulu ke web server dan dibuka menggunakan WAP browser melalui perangkat.

Pada tahap pembuatan, untuk lebih mempermudah pembuatan file WML ini dapat digunakan WAP simulator yang mampu melakukan simulasi tampilan file WML secara lokal pada komputer tanpa harus melakukan upload terlebih dahulu. Contoh WAP Simulator antara lain UPSDK, Wireless Companion, dan lain-lain. Setelah disain telah sesuai dengan keinginan maka file WML baru di-upload ke server sehingga dapat diakses secara luas melalui internet. Selain format WML, beberapa format yang juga cukup banyak digunakan adalah xHTML dan cHTML.



Gambar X-5 WAP simulator

## X.7 Pengembangan Aplikasi untuk Konten

Untuk lebih memahami pengembangan aplikasi pembelajaran terutama yang berbasis Java, berikut ini akan diberikan contoh source code aplikasi sederhana. Aplikasi ini masih perlu dikembangkan lebih lanjut untuk dapat dipakai sebagai aplikasi pembelajaran.

Pada beberapa contoh dibawah ini dilakukan tiga proses yang cukup lazim dilakukan pada aplikasi *mLearning*. Proses tersebut adalah membaca file lokal dan *remote*. Proses-proses yang melibatkan koneksi jaringan memiliki prosedur yang sama seperti melakukan koneksi jaringan pada perangkat *mobile* yang telah dibahas di bab V.

Konten pembelajaran dapat berupa sebuah file lokal yang ada dalam perangkat dan biasanya disertakan dalam distribusi aplikasi. Berikut ini contoh aplikasi yang mampu membaca konten yang tersimpan dalam sebuah file teks lokal bernama **mycontent.txt**.

**Listing X-1** MIDlet untuk membaca file local (`ReadFile.java`)

---

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*; import
java.io.*;

public class ReadFile extends MIDlet implements CommandListener {
private Form mnForm;      private Command exitCommand;

public ReadFile() {
    mnForm = new Form("");
    exitCommand = new Command( "Exit", Command.EXIT, 1 );
    mnForm.addCommand(exitCommand);
    mnForm.setCommandListener(this);
}
public void startApp() { mnForm.append(new StringItem("My
Content\n", readFileText("mycontent.txt")));
Display.getDisplay(this).setCurrent(mnForm);
}//end startApp()

public void pauseApp() {
}//end pauseApp()

public void destroyApp( boolean unconditional ) {
}//end destroyApp

public void commandAction( Command com, Displayable dis ) {
if ( com == exitCommand ) {           destroyApp( true );
    notifyDestroyed();
}//end
    CommandAction( Command, Displayable )

private String readFileText(String ftext){
    InputStream is = getClass().getResourceAsStream(ftext);
try{
    StringBuffer sb = new StringBuffer();
    int chr, i = 0;
    // Read until the end of the stream
    while ((chr = is.read()) != -1)
        sb.append((char) chr);           return
sb.toString();
}catch (Exception e){
    System.out.println("Unable to create stream");
}
return
null;
}
}//end MyMidlet
```

---

Konten pembelajaran akan lebih mudah diakses secara luas jika disimpan di internet. J2ME memiliki kemampuan membaca remote file dari web server di internet. berikut ini contoh source code J2ME yang memiliki kemampuan membaca dan menampilkan isi sebuah konten yang disimpan di sebuah remote file.

Diasumsikan konten pembelajaran disimpan dalam file teks bernama **mycontent.txt** dan berada di server web **www.myserver.com**.

**Listing X-2** Membaca file yang ada di server web (`httpConnection.java`)

---

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.io.*; import
java.io.*;

public class httpConnection extends MIDlet implements CommandListener {
private Command exit, start; private Display display; private Form
form;

    public httpConnection (){    display =
Display.getDisplay(this);    exit = new
Command("Exit", Command.EXIT, 1);        start = new
Command("Start", Command.EXIT, 1);        form = new
Form("Http Connection");        form.addCommand(exit);
    form.addCommand(start);
    form.setCommandListener(this);
}
public void startApp(){
    display.setCurrent(form);
}
public void pauseApp() {}

public void destroyApp(boolean unconditional){
destroyApp(false);    notifyDestroyed();
}
public void commandAction(Command command, Displayable displayable){
if (command == exit){    destroyApp(false);    notifyDestroyed();
}else if (command == start){
    HttpConnection connection = null;
    InputStream inputstream = null;
try{
    connection = (HttpConnection)
    Connector.open("http://www.myserver.com/mycontent.txt");
    //HTTP Request
    connection.setRequestMethod(HttpConnection.GET);
connection.setRequestProperty("Content-Type","//text plain");
connection.setRequestProperty("Connection", "close");
    // HTTP Response
    System.out.println("Status Line Code: " +
    connection.getResponseCode());
    System.out.println("Status Line Message: " +
    connection.getResponseMessage());
    if (connection.getResponseCode() == HttpConnection.HTTP_OK){
System.out.println(
    connection.getHeaderField(0)+" "+connection.getHeaderFieldKey(0));
    System.out.println("Header           Field           Date:      "+
connection.getHeaderField("date"));        String str;
```

```
    inputstream = connection.openInputStream();
int length = (int) connection.getLength();                      if
(length != -1){
    byte incomingData[] = new byte[length];
inputstream.read(incomingData);           str = new
String(incomingData);
}else{
    ByteArrayOutputStream bytestream = new ByteArrayOutputStream();
    int ch;
    while ((ch = inputstream.read()) != -1){
        bytestream.write(ch);
    }
    str = new String(bytestream.toByteArray());
    bytestream.close();
}
System.out.println(str);
}
}catch(IOException error){
    System.out.println("Caught IOException: " + error.toString());
}finally{
    if (inputstream!= null){
try{
    inputstream.close();
}catch( Exception error){ /*log error*/}
}
if (connection != null){
try{
    connection.close();
}catch( Exception error){/*log error*/}
}
}
}
}
```

# XI Aplikasi Mobile Learning

## XI.1 Beberapa Aplikasi m-Learning yang Ada

Saat ini relatif belum banyak penelitian ataupun produk *m-Learning* yang muncul di pasaran dibandingkan dengan penelitian dan produk *e-Learning*. Penelitian ataupun produk *m-Learning* yang ada dikembangkan saat ini antara lain adalah sebagai berikut:

- a. StudyTXT
  - b. Sussex Mobile Interactive Learning Environment (SMILE)
  - c. University of North Carolina at Wilmington Mobile Learning Environment (UNCW MLE)
  - d. MOBIlearn, ObjectJ mLearning, *Mobile Learning Engine* (MLE)

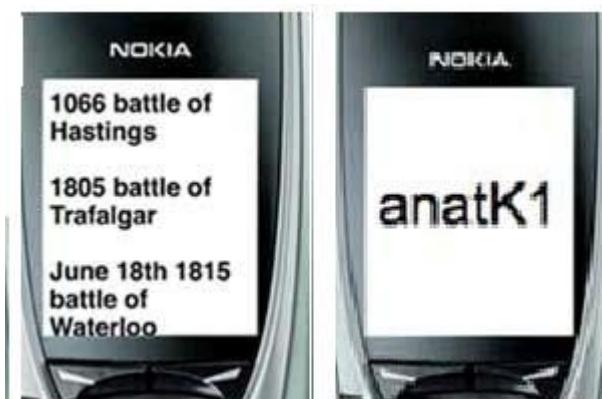
- e. Microsoft Education Pack for Tablet PC
- f. Ganesha Mobile Learning (GML).

## XI.2 StudyTXT

StudyTXT merupakan sebuah pendukung *study on demand*, sebuah program di Auckland University of Technology bekerjasama dengan Telecom dan Vodafone dengan menggunakan pesan teks pada telepon seluler. Program ini memulai operasional pada Oktober 2005 dan diklaim sebagai *mobile study-support* pertama di dunia dan dinilai cukup sukses. StudyTXT dirancang untuk menyediakan layanan *m-Learning* nasional bagi institusi pendidikan di Selandia Baru. Program ini juga masuk dalam salah satu finalis dalam The Computerworld Excellence Awards 2006 di Selandia Baru.

Skenario StudyTXT adalah pengguna StudyTXT yang memanfaatkan layanan ini mengirimkan teks singkat berisi kode tertentu menggunakan pesan SMS ke server StudyTXT. Server kemudian merespon permintaan ini dengan mengirimkan SMS berisi informasi singkat tentang obyek yang dipilih. Pesan SMS ini kemudian disimpan dalam perangkat seluler siswa dan dapat di-review kapan saja dan dimana saja, ini merupakan catatan tradisional dalam pembelajaran yang menyenangkan yang dapat didapat dalam waktu singkat dalam bentuk SMS.

StudyTXT merupakan layanan yang cukup mudah. Pesan SMS dapat di-*download* untuk tiap pesan SMS atau tiap subyek/topik yang berisi beberapa SMS. Sebagai contoh, sebuah subyek "*Lower leg muscles*" (dalam pelajaran anatomii) memiliki tiga buah pesan. Jika ingin *download* semua pesan maka pengguna harus mengirimkan pesan SMS berisi kode "anatK" ke nomor 396 dan pengguna akan mendapatkan tiga pesan SMS. Sedangkan untuk *download* satu pesan SMS saja, misalnya pesan nomor 2, maka kode yang harus dikirimkan adalah "anatK2". Biaya SMS dengan cara pertama adalah 0,3 sen dan untuk cara kedua adalah 0,5 sen. Selain itu pengguna juga dapat berlangganan 500 pesan SMS dengan biaya \$10. Trafik StudyTXT akan sangat padat pada akhir minggu, terkadang ada penundaan (*delay*) satu sampai dua jam.



#### **GambarXI-1** Sesi pembelajaran StudyTXT menggunakan SMS

Dengan adanya keterbatasan jumlah karakter dalam setiap pesan, maka kualitas konten sangat diperhatikan. Konten pembelajaran dibuat secara khusus oleh ahli yang kompeten dan dimoderasi sehingga dapat lebih akurat dan tepat sasaran. Tanggapan dari siswa maupun akademisi sangat positif sehingga jumlah pesan SMS yang *di-download* telah mendekati jumlah yang dibutuhkan proyek ini untuk dapat secara mandiri membiayai pendanaannya sendiri.

### **XI.3 SMILE**

SMILE adalah program permulaan untuk memperkenalkan XDA (perangkat yang mempunyai kemampuan fungsional seperti pada PDA dan penuh dengan akses internet) ke dalam kurikulum pada The School of Cognitive and Computer Sciences di University of Sussex, United Kingdom. SMILE mulai dibangun pada tahun 2001 dan menjadi lengkap dengan perangkat *mobile* pada tahun 2003 untuk mahasiswa yang mengambil pelatihan selama tiga tahun. SMILE menggunakan delapan XDA yang digunakan sebagai bagian pelatihan pada lingkungan pembelajaran yang interaktif. Koneksi akses internet dengan menggunakan GPRS. Materi pembelajaran disediakan dalam bentuk *word processing*, *email*, *web* dan kertas. Pemberian materi pembelajaran dilakukan dalam pertemuan tatap muka dan secara *online*. Dibentuk kelompok-kelompok kecil untuk berkolaborasi dengan berdiskusi secara bertatap muka maupun *online*.

### **XI.4 UNCW MLE**

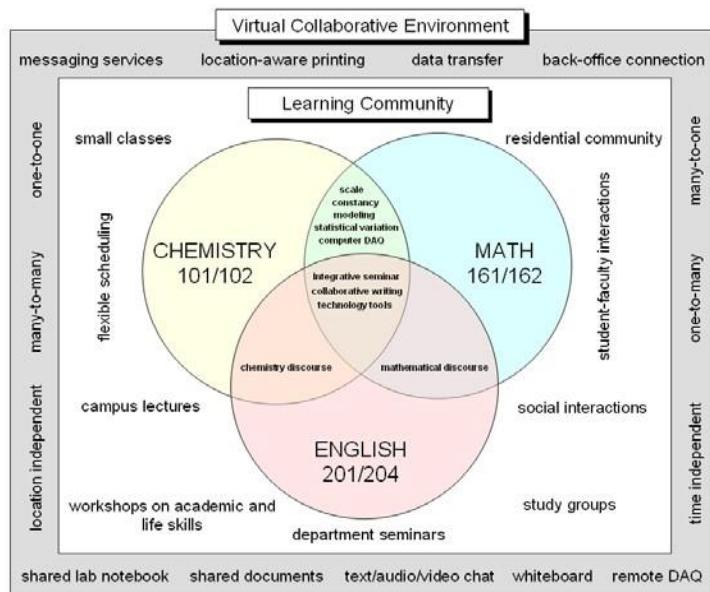
UNCW MLE adalah proyek *m-Learning* yang menggunakan PDA sebagai perangkat *client*. Desain menu tergambar seperti pada gambar XI-2. UNCW MLE dibangun dengan tujuan :

- a. Untuk membangun lingkungan yang mudah digunakan untuk berbagai perangkat dan *platform*.
- b. Untuk menyediakan integrasi yang komprehensif dari komunikasi kolaborasi dan komponen *computing*.
- c. Memperbaiki pertukaran data dengan menggunakan *web service*.

UNCW MLE mempunyai tiga pelatihan yaitu bahasa inggris, kimia dan matematika. Arsitektur UNCW MLE terlihat pada Gambar berikut.



**Gambar XI-2** Menu pada Antarmuka UNCW MLE



**Gambar XI-3** Lingkungan Virtual yang Kolaboratif

## XI.5 MOBILearn

MOBILearn adalah proyek penelitian di University of Birmingham yang difokuskan pada paradigma generasi berikutnya dan antarmuka yang mendukung teknologi pembelajaran dalam lingkungan peralatan *mobile* yang terbatas dan mempunyai potensi kecerdasan *ambient*. *Ambient* adalah pandangan bahwa teknologi akan sensitif terhadap kebutuhan pengguna dan kebiasaan pengguna, sehingga akan menyediakan informasi yang sangat berkualitas dan tersedia untuk banyak pengguna, kapanpun, dimanapun dan pada perangkat apapun. MOBILearn dibangun berdasarkan empat skenario berdasarkan penggunanya, yaitu sebagai berikut:

- Pembelajaran ditujukan untuk kebutuhan dari anggota Tim Pertolongan Pertama di Open

University

- b. Pembelajaran melibatkan mahasiswa yang baru masuk
- c. Pembelajaran ditujukan pada mahasiswa yang mengambil mata kuliah Sejarah Eropa abad 14 – 15 di Universita Cattolica dan Uffizzi
- d. Pembelajaran yang melibatkan manajer finansial yang mengambil kuliah eksekutif *Master Bachelor of Art (MBA)* yang mempunyai waktu kuliah dua hari di University of Zurich

Skenario penghantaran konten untuk mahasiswa MBA adalah jika mahasiswa sedang berada dalam perjalanan dengan kereta, mereka dapat mengerjakan soal kuis yang di-*download* ke perangkat *mobile* mereka. Mahasiswa yang mengambil mata kuliah sejarah dapat mendapatkan materi tentang suatu benda seni kapanpun, sewaktu dia tepat berada di depan atau di sekitar benda seni tersebut. Sedangkan skenario penggunaan untuk anggota Tim Pertolongan Pertama adalah pada saat anggota tim berada di lapangan dapat mendapatkan instruksi langsung dengan sebuah alat yang dapat merespon untuk mengatasi kecelakaan yang parah dan jauh dari fasilitas kesehatan.

## XI.6 ObjectJ mLearning

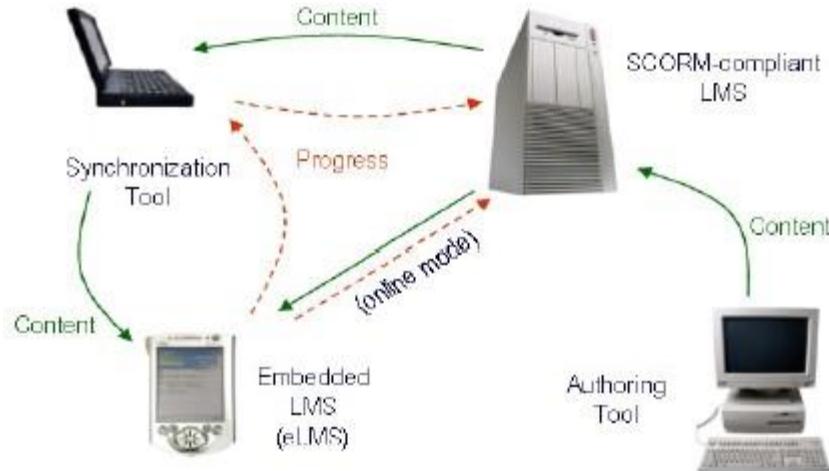
Object mLearning adalah sebuah produk *m-Learning* dari ObjectJ Inc., sebuah perusahaan yang memfokuskan pada penelitian dan pengembangan solusi bagi industri *e-Learning*. Aplikasi ObjectJ mLearning mampu diintegrasikan dengan ObjectJ eLearning dan memiliki fitur-fitur berikut :

- a. Mendukung penggunaan PocketPC sebagai klien *e-Learning*
- b. Model konten yang dapat disinkronisasi secara interaktif
- c. Mendukung komponen Macromedia Flash sebaik dukungan konten *Hyper Text Markup Language* (HTML)
- d. Mendukung beberapa bentuk ujian (pilihan ganda, jawaban singkat)
- e. Menyertakan *mLearning Commander* yaitu sebuah perangkat sinkronisasi
- f. *SCORM compliant*
- g. Mendukung komponen video dalam waktu dekat

Adapun domain dan teknologi yang dipakai dalam pengembangan ObjectJ mLearning adalah sebagai berikut :

- a. Desain dan pengembangan aplikasi menggunakan J2EE
- b. Rekayasa aplikasi menggunakan UML
- c. Teknologi *m-Learning* menggunakan J2ME (*MIDP Palm OS Profile*)
- d. Integrasi dengan aplikasi Java dan non-Java di lingkungan J2EE menggunakan teknologi *Java Native Interface* dan COM

- e. Pengembangan aplikasi multimedia menggunakan teknologi *Java Media Framework* dan *ActiveX*
- f. Pengembangan *e-course* (khususnya menggunakan Flash, JavaScript dan multimedia)



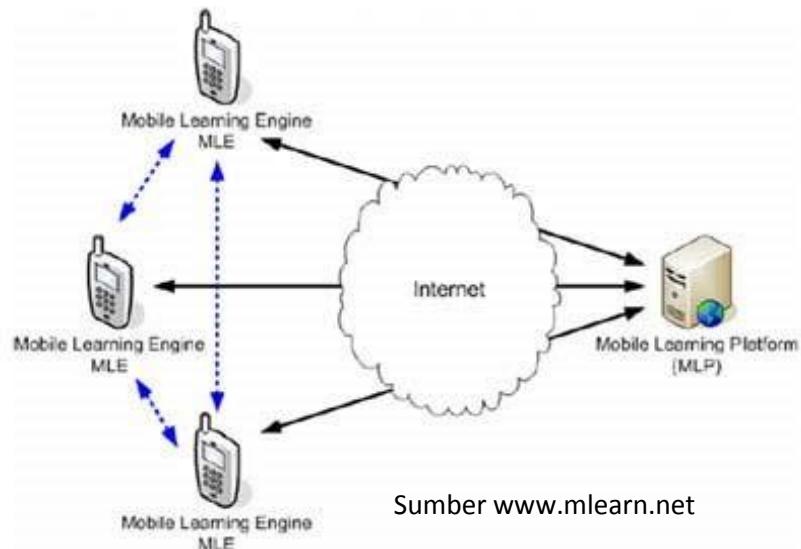
**Gambar XI-4** Arsitektur Pembelajaran ObjectJ

## XI.7 MLE

MLE adalah sebuah aplikasi *m-Learning* yang komprehensif yang dibangun dengan J2ME. MLE memindahkan *computer aid* dan *multimedia based-learning* (yang dikenal dengan *e-Learning*) ke lingkungan perangkat *mobile* yang dikenal dengan *m-Learning*. Bersama MLE pelajar dapat belajar dimanapun dan kapanpun dengan menggunakan *smartphone*. MLE sangat memperlihatkan perkembangan dan potensi teknologi *m-Learning*, karenanya MLE telah mendapatkan beberapa penghargaan dalam perlombaan di negaranya maupun di benua Eropa. MLE dinominasikan dalam *the Multimedia Transfer 2005 award* dan menjadi pemenang pada *The International e-Learning Fair Learntec 2005* di Karlsruhe Jerman. MLE juga menjadi pemenang *the "Sonderpreis"* pada *the L@rnie Award 2005*. Pemenang pada *The European Top Talent Award 2004* dalam kategori "Mobile Contents". Pemenang pada *Austrian Innovation Award 2004* dalam kategori "Mobile Applications".

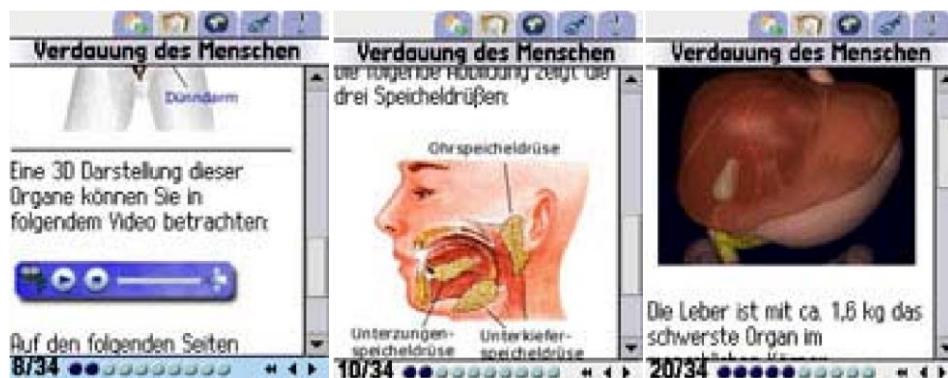
MLE digunakan dengan sebuah *learning-platform*. *Learning- platform* adalah sebuah *server* pada internet yang menyediakan materi pembelajaran dan beberapa tambahan fungsi lainnya seperti *hypermedia-systems* (contohnya glosarry atau jaringan pengetahuan) dan komunikasi (contohnya forum multimedia berisi rekaman multimedia ataupun rekaman suara yang langsung direkam dengan smartphone yang digunakan). MLE berkomunikasi dengan *mobile learning platform* (MLP) melalui *Hyper Text Transport Protocol* (HTTP) dan *eXtensible Markup Languange* (XML). Keduanya merupakan standar internasional

sehingga memungkinkan untuk menggunakan *Learning Management Systems* (LMS) yang sudah ada sebagai *learning-platform*. Obyek pembelajaran dibangun dengan XML. Dengan materi pembelajaran pembelajar dapat menggunakan teks, gambar, suara dan elemen video sehingga menjadi seperti materi yang interaktif.



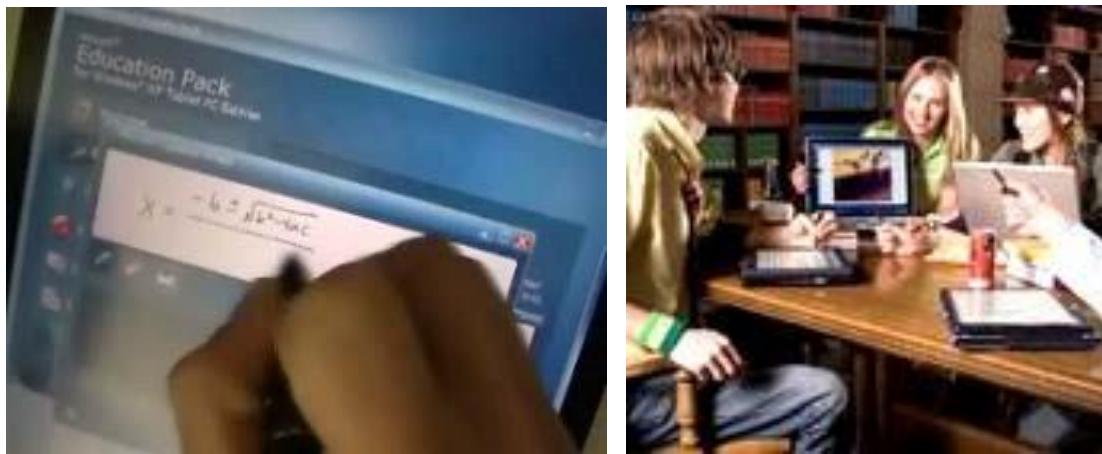
Sumber [www.mlearn.net](http://www.mlearn.net)

Gambar XI-5 Komunikasi MLE dengan MLP



Gambar XI-6 Materi pembelajaran MLE

## XI.8 Microsoft Education Pack for tablet PC



**Gambar XI-7** Penggunaan *Microsoft Education Pack for Tablet PC* untuk pembelajaran

*Microsoft Education Pack for Tablet PC* adalah aplikasi gratis yang diluncurkan Microsoft untuk pelajar pada saat di sekolah dengan perangkat yang digunakan adalah tablet PC. Aplikasi ini terdiri dari lima program yang dapat membantu pelajar belajar dengan terorganisasi dan efektif. Pelajar juga dapat merasakan kesenangan dalam belajar dengan menggunakan aplikasi ini. Aplikasi ini dapat di-download pada alamat web site <http://www.microsoft.com/downloads/details.aspx?FamilyId=9D346916-B52687E-19D0BCE568DEF39&displaylang=en>.

## XI.9 GML

Ganesa Mobile Learning atau GML adalah prototipe aplikasi m-learning yang sedang dikembangkan pada sebuah penelitian di Institut Teknologi Bandung (ITB). Pada penelitian ini teknologi pengembangan menggunakan Java (J2EE dan J2ME). Perangkat yang digunakan adalah Nokia Series 60 (Nokia 6600) dan pada saat ini sedang dikembangkan versi aplikasi yang sesuai dengan merek dan tipe perangkat lain. Beberapa fitur GML sudah cukup stabil sementara fitur lain masih dalam penelitian lebih lanjut.

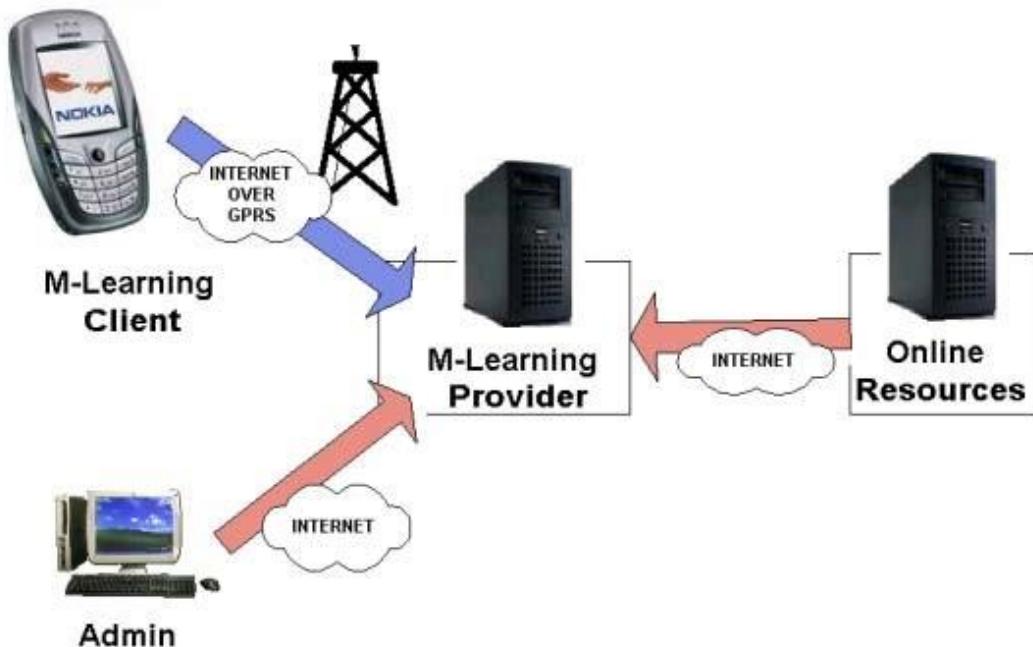
Pengembangan sistem GML diarahkan pada dua sisi yaitu sebagai berikut :

- a. Mengembangkan GML Provider yang merupakan sebuah aplikasi layanan m-learning berbasis web di sisi server,
- b. Mengembangkan GML Client yang merupakan aplikasi mobile di sisi client yang dapat mengakses layanan GML Provider.

Secara umum GML memiliki kemampuan sebagai berikut:

- a. Menyediakan fasilitas untuk melakukan fungsi administratif pembelajaran, meliputi registrasi, autentikasi dan otorisasi
- b. Menyediakan fasilitas untuk melakukan pengelolaan, pengaksesan dan penghantaran materi pembelajaran meliputi modul belajar, materi tambahan pengajar, penugasan, kuis latihan dan ujian
- c. Menyediakan fasilitas untuk melakukan interaksi dan komunikasi secara *asynchronous* antara peserta ajar dengan pengajar maupun antar sesama peserta
- d. Menyediakan fasilitas untuk melakukan pengaksesan perangkat bantu produktifitas pembelajaran meliputi pencarian definisi istilah pada server GML dan pencarian definisi istilah pada *remote server*

Arsitektur fisik GML tergambar seperti pada Gambar 3.4. Pertukaran pesan antara GML *Provider* dan GML *Client* menggunakan format XML dan protokol *messaging* XML-RPC, sedangkan transport data menggunakan protokol HTTP over GPRS. GML Client dibagi menjadi dua bagian, yaitu GML Coach yang ditujukan untuk pengajar serta GML Learner yang ditujukan bagi pembelajar/siswa.



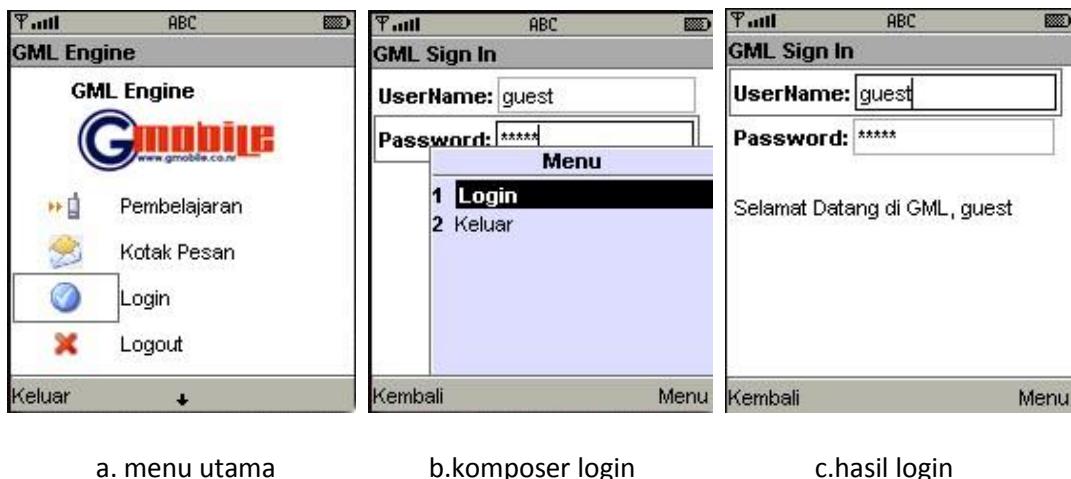
**Gambar XI-8** Arsitektur Fisik GML

Fitur-fitur yang ada pada GML Learner yang meliputi:

- a. Login
- b. Browsing modul online
- c. Download modul
- d. Melihat modul pada penyimpanan lokal perangkat seluler
- e. *Hypermedia*
- f. Ruang pembelajaran yang meliputi forum, ujian, materi pengajar, FAQ dan tugas
- g. Pengaturan
- h. Profil

Login dilakukan untuk mendapatkan sessionID dari server yang diperlukan sebagai salah satu parameter jika pengguna ingin koneksi ke server untuk fitur yang lainnya. Langkahnya adalah sebagai berikut :

- a. Pada menu utama (a) pilih tombol login
- b. Isi username dan password pada komposer login (b) kemudian pilih perintah login pada menu pulldown
- c. Jika login diterima akan terlihat hasil autentikasi (c)



a. menu utama

b.komposer login

c.hasil login

**Gambar XI-9** Tampilan login GML

Browsing modul adalah melihat isi modul dari mata ajar yang diikuti secara online dari server.

Langkahnya adalah sebagai berikut :

- a. Pada menu utama (a) pilih tombol pembelajaran

- b. Pilih tombol Mata Ajar di Server pada menu pembelajar (b)
- c. Sorot salah satu mata ajar yang disediakan dan pada menu pulldown pilih perintah Lanjut (c)
- d. Sorot salah satu kelas mata ajar yang dibuka dan pada menu pulldown pilih perintah lanjut (d)
- e. Sorot salah satu modul dan pada menu pulldown pilih perintah Browse (e)
- f. Untuk melakukan navigasi pada tampilan isi modul (f) dapat memilih perintah Berikut atau Sebelum. Perintah Berikut untuk menuju halaman berikutnya. Perintah Sebelum untuk menampilkan halaman selanjutnya



a.menu utama

b.menu pembelajar

c.daftar mata ajar



d.daftar kelas yang dibuka

e. daftar modul

e.isi modul

**Gambar XI-10** *Browsing* modul pada GML

*Download* modul adalah mendownload isi modul dari mata ajar yang diikuti untuk disimpan di penyimpanan lokal perangkat seluler. Langkahnya adalah sebagai berikut :

- a. Pada menu utama (a) pilih tombol pembelajaran
- b. Pilih tombol Mata Ajar di Server pada menu pembelajar (b)
- c. Sorot salah satu mata ajar yang disediakan dan pada menu pulldown pilih perintah Lanjut (c)

- d. Sorot salah satu kelas mata ajar yang dibuka dan pada menu pulldown pilih perintah lanjut (d)
- e. Sorot salah satu modul dan pada menu pulldown pilih perintah *Download* (e)
- f. Selama proses pen-*download*-an akan tampil tampilan tunggu. Silahkan tunggu sampai tampilan tunggu selesai, kemudian modul yang telah didownload dapat dibuka tanpa harus terhubung dengan server (f).



a. menu utama

b. menu pembelajar

c. daftar mata ajar



d. daftar kelas yang dibuka

e. daftar modul

f. tampilan tunggu

**Gambar XI-11** Tampilan *download* pada GML

Melihat modul pada penyimpanan local perangkat seluler adalah melihat isi modul dari mata ajar yang tersimpan pada penyimpanan lokal perangkat seluler. Pembelajar dapat belajar secara *offline* tanpa melakukan koneksi dengan server. Langkahnya adalah sebagai berikut :

- a. Pada menu utama (a) pilih tombol pembelajaran
- b. Pilih tombol Mata Ajar di Divais pada menu pembelajar (b)

- c. Sorot salah satu kelas mata ajar yang ada dan pada menu pulldown pilih perintah Lanjut (c)
- d. Sorot salah satu modul ajar yang ada dan pada menu pulldown pilih perintah lanjut (d)
- e. Pembelajar dapat melihat isi modul (e)
- f. Pada tampilan isi modul terdapat beberapa perintah menu pulldown, yaitu :
- g. Berikut, untuk melihat halaman berikutnya
- h. Sebelum, untuk melihat halaman sebelumnya
- i. Quiz, untuk melakukan quiz dengan melakukan koneksi ke server
- j. *Glossary*, untuk mendapatkan deskripsi istilah yang telah diisikan dalam textbox dari *glossary* server
- k. YahooEnsiklopedia, untuk mendapatkan deskripsi istilah yang telah diisikan dalam textbox dari server YahooEnsiklopedia



a.menu utama



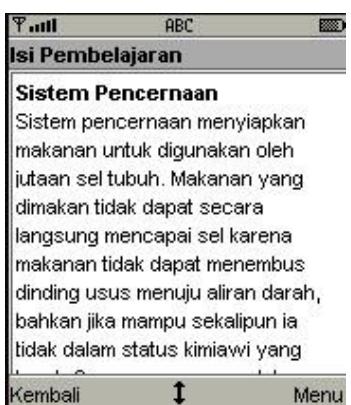
b.menu pembelajar



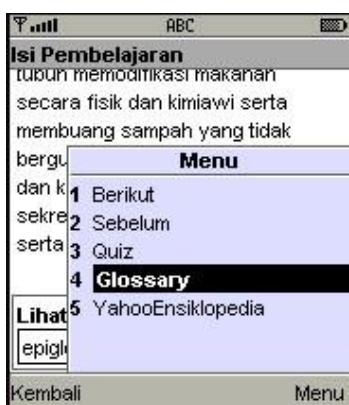
c.daftar kelas mata ajar



d.daftar modul



e. isi modul



e.menu pulldown modul **Gambar XI-**

## 12 Browsing modul-modul dalam GML

*Hypermedia* adalah mencari deskripsi istilah yang tidak dimengerti ke Glossary Server GML atau ke Server Ensiklopedia Yahoo. Langkahnya adalah sebagai berikut :

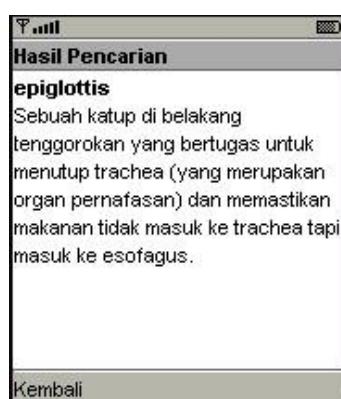
- a. Pada menu utama (a) pilih tombol pembelajaran
- b. Pilih tombol Pencarian Deskripsi Istilah pada menu pembelajar (b)
- c. Isikan istilah yang ingin dicari pada textfiel istilah (c)
- d. Pilih perintah Glossary Server untuk mencari deskripsi istilah tersebut pada glossary server GML.
- e. Pilih perintah Ensiklopedia Yahoo pada menu *pull down* untuk mencari deskripsi istilah tersebut pada server Ensiklopedia Yahoo.
- f. Tampilan hasil pencarian (d)



a.menu utama

b.menu pembelajar

c.komposer pencarian istilah



d.hasil pencarian

**Gambar XI-13** Tampilan *hypermedia* pada GML

Ruang pembelajaran adalah fitur yang disediakan supaya pembelajar dapat melakukan proses pembelajaran secara *online*. Fasilitas yang disediakan adalah :

- a. Ujian, pembelajar dapat melakukan ujian secara online
- b. Tugas, pembelajar dapat mengetahui tugas dan mengirim tugas tersebut
- c. Materi pengajar, pembelajar dapat melihat materi tambahan dari pengajar
- d. Tanya jawab, pembelajar dapat mengirim pertanyaan pada pengajar dan dapat menlihat kumpulan tanya jawab yang ada
- e. Forum, pembelajar dapat menyalurkan opininya dengan mengirim pesan forum untuk menanggapi topik forum yang ada

Berikut langkah untuk mengirimkan opini pada forum :

- a. Pada menu Pembelajar pilih tombol Ruang Pembelajaran (a)
- b. Sorot salah satu kelas mata ajar yang ada dan pada menu pulldown pilih perintah Forum (b)
- c. Pilih tombol Forum pada menu forum (c)
- d. Sorot salah satu topik forum dan pilih perintah Lanjut (d)
- e. Pilih perintah Tanggapi pada menu *pull down* pada tampilan daftar pesan forum (e).
- f. Isikan Opini dala textfield dan pilih menu Kirim pada menu *pull down*, maka opini akan terkirim ke server



- a. menu pembelajar
- b. komposer pencarian istilah
- c. menu foun



d.daftar topik forum e. daftar pesan forum e.komposer forum

**Gambar XI-14** Tampilan ruang pembelajaran pada GML

Pengaturan/*setting* adalah fasilitas untuk menghapus modul yang tersimpan dalam penyimpanan lokal perangkat seluler. Langkahnya adalah sebagai berikut :

- Pada menu utama (a) pilih tombol pembelajaran
- Pilih tombol Pengaturan pada menu pembelajar (b)
- Pilih mata ajar yang semua modulnya akan dihapus dari penyimpanan local perangkat seluler. Pilih perintah Hapus pada menu pulldown.



a.menu utama

b.menu pembelajar

c. pengaturan

**Gambar XI-15** Tampilan pengaturan pada GML

Profil adalah fasilitas untuk melihat profil pembelajar. Langkahnya adalah sebagai berikut :

- Pada menu utama (a) pilih tombol pembelajaran
- Pilih tombol Profil pada menu pembelajar (b)

c. Profil pembelajar (c), yaitu profil pebelajar yang tersimpan di server



**Gambar XI-16** Tampilan profil pada GML



# Daftar Pustaka

---

Attewell, Jill, dan Smith, C. Savill, *Mobile Technologies and Learning*, London, 2005,  
<http://www.LSDA.org.UK>, 18 Februari 2006 15.20 WIB

Bloch, Cynthia dan Annette Wagner, "MIDP 2.0 Style Guide for the Java 2 Platform Micro Edition", Addison Wesley, 2003

Fox, David dan Roman Verhosek, "Micro java Game Development", Addison Wesley, 2002

Georgiev, T., E. Georgieva, A. Smrikarov (2005), *A General Classification of Mobile Learning Systems*, International Conference on Computer Systems and Technologies - CompSysTech',  
<http://www.ecet.ecs.ru.acad.bg/cst05/Docs/cp/sIV/IV.14.pdf>, 9 Agustus 2006, 15.46 WIB.

Meisenberger, Matthias dan Alexander K.Nischelwitzer, *The mobile learning engine (MLE) - a mobile, computer-aided, multimedia-based learning application*, Multimedia Application in Education Conference,

[http://www.drei.fhjoanneum.at/mle/docs/Matthias\\_Meisenberger\\_MApEC\\_Paper\\_mLearning.pdf](http://www.drei.fhjoanneum.at/mle/docs/Matthias_Meisenberger_MApEC_Paper_mLearning.pdf), 9 Agustus 2006, 14.03 WIB.

Riggs, Roger, Antero Taivalsaari et al, "Programming Wireless Devices with Java 2 Platform Micro Edition, Second Edition", Addison Wesley, 2003

Sun Microsystems, "Java Technology (J2ME)", <http://java.sun.com/javame>, 20 Februari 2007

Wells, Martin J., "J2ME Game Programming", Premier Press, 2004

Wood, Karen (2003), *Introduction to Mobile Learning (M-Learning)*, Ferl, Becta (British Educational Communications and Technology Agency),  
<http://www.ferl.becta.org.uk/display.cfm?page=65&catid=192&resid=5194>, 19 Nopember 2005 15.20 WIB

# Lampiran

---

## Sun Java Wireless Toolkit

Sebelum mulai membuat aplikasi *mobile*, terlebih dahulu dipersiapkan seperangkat tool yang diperlukan selama fase pengembangan. Tool pengembangan yang dijelaskan pada bagian ini adalah Sun Java Wireless Toolkit, yang tersedia gratis di web Sun Microsystems pada alamat <http://java.sun.com>. Sun Java Wireless Toolkit merupakan seperangkat tool yang ditujukan bagi para pembuat aplikasi di dalam mengembangkan aplikasi-aplikasi *mobile* untuk telepon seluler ataupun perangkat nirkabel lainnya. Versi terbaru dari Sun Java Wireless Toolkit pada saat buku ini ditulis adalah 2.5.

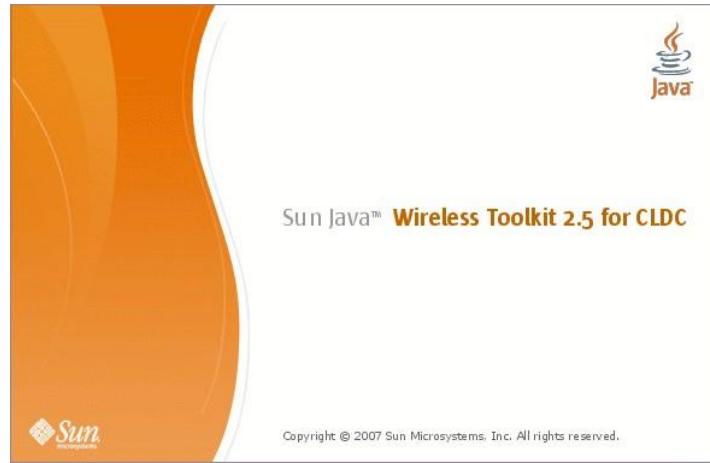
Sun Java Wireless Toolkit memiliki tiga komponen utama sebagai berikut.

- a. KToolbar, merupakan komponen yang berfungsi menyederhanakan berbagai aktivitas di dalam mengembangkan aplikasi *mobile*, seperti menjalankan emulator dan berbagai program utilitas, membuat paket instalasi aplikasi dalam format .jar dan .jad, menyediakan fasilitas *build* untuk melakukan kompilasi dan verifikasi kode program, dan lain sebagainya.
- b. Emulator, merupakan komponen yang merepresentasikan perangkat *mobile* guna menyediakan lingkungan simulasi aplikasi sebelum diinstalasi ke perangkat *mobile* yang sebenarnya.
- c. Program-program utilitas yang menyediakan berbagai fungsionalitas pendukung, seperti *memory monitor*, *profiler*, dan *text messaging console*.

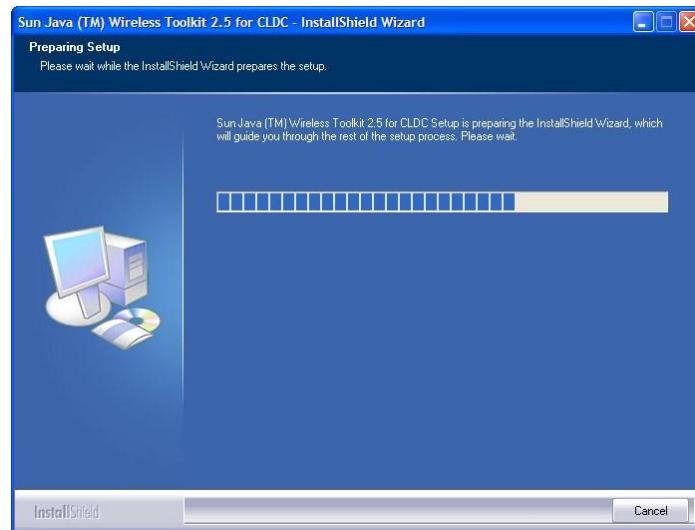
Tool lain yang diperlukan selama fase pengembangan aplikasi *mobile* adalah J2SE Development Kit dan teks editor atau Java IDE yang digunakan untuk menulis atau mengedit kode program Java. Sebagai catatan, untuk penggunaan Sun Java Wireless Toolkit versi 2.5, versi minimal dari J2SE Development Kit yang diperlukan adalah 1.5.

Berikut ini dijelaskan langkah-langkah melakukan instalasi Sun Java Wireless Toolkit 2.5 pada sistem operasi Windows.

Sebelum mulai melakukan instalasi Sun Java Wireless Toolkit 2.5, perlu dipastikan J2SE Development Kit versi 1.5 atau versi lebih baru telah diinstalasi. Jika J2SE Development Kit yang dimaksud telah diinstalasi, proses instalasi Sun Java Wireless Toolkit 2.5 dapat dilakukan. Tampilan awal proses ini dapat dilihat pada gambar berikut.

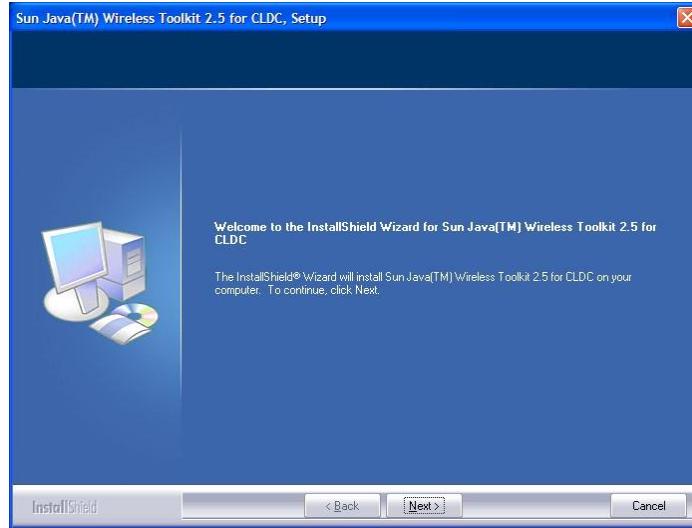


**Gambar A-1a** Tampilan Awal Instalasi Sun Java Wireless Toolkit



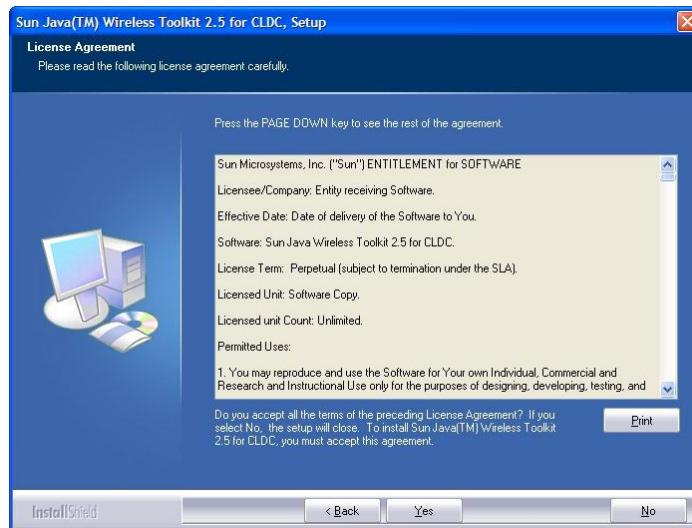
**Gambar A-1b** Tampilan Awal Instalasi Sun Java Wireless Toolkit

1. Tampilan selanjutnya memuat permintaan konfirmasi apakah instalasi akan dilanjutkan atau tidak. Instalasi akan dilanjutkan ketika tombol Next dipilih.

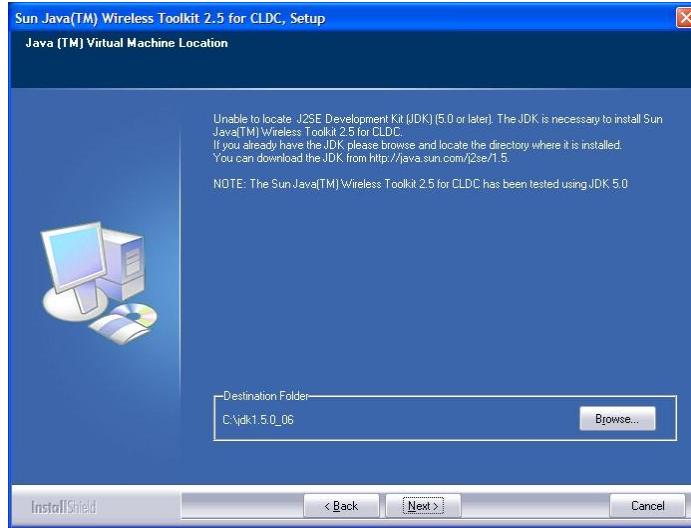


**Gambar A-2** Tampilan Konfirmasi Untuk Melanjutkan Instalasi

2. Tampilan selanjutnya memuat informasi *license agreement*, seperti yang dapat dilihat pada Gambar A-3. Proses instalasi akan dilanjutkan ketika *license agreement* disetujui dengan memilih tombol Yes.
3. Proses selanjutnya adalah menentukan lokasi di mana J2SE Development Kit diinstalasi. Proses ini pada dasarnya dilakukan secara otomatis oleh program instalasi, tetapi terdapat kemungkinan program tidak dapat menemukan lokasi yang dimaksud. Lokasi yang dimaksud dapat ditentukan secara manual dengan memilih tombol Browse. Jika telah selesai, proses dapat dilanjutkan dengan memilih tombol Next. Proses ini diilustrasikan pada Gambar A-4.

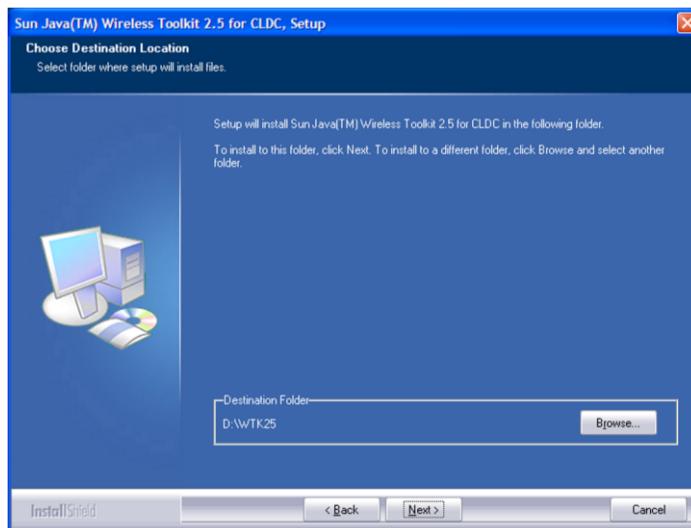


**Gambar A-3** Tampilan *License Agreement*



**Gambar A-4** Tampilan Lokasi Instalasi J2SE Development Kit

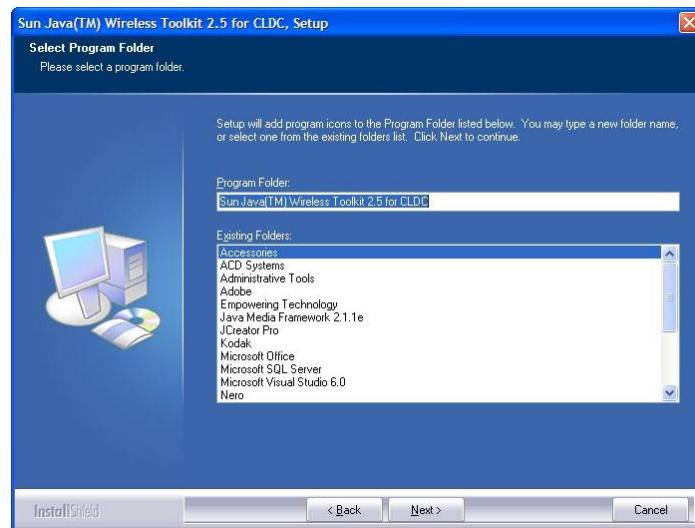
4. Proses instalasi dilanjutkan dengan memilih lokasi instalasi Sun Java Wireless Toolkit. Lokasi ini dapat mengikuti lokasi default yang diberikan pada saat instalasi.



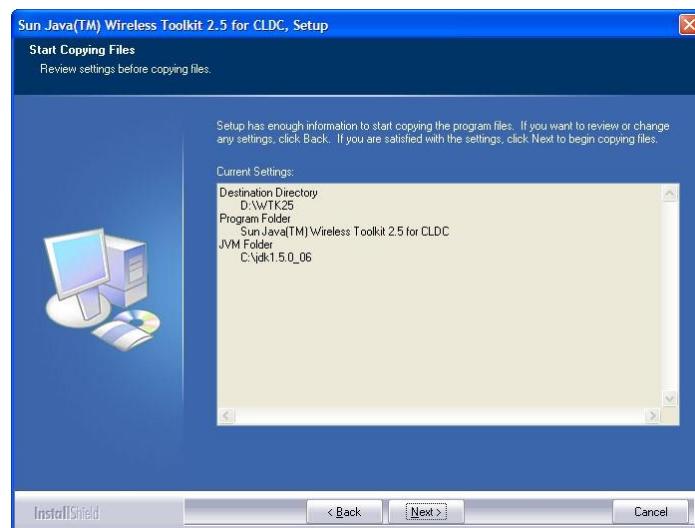
**Gambar A-5** Tampilan Pemilihan Lokasi Instalasi Sun Java Wireless Toolkit

5. Tampilan folder program akan diperlihatkan segera setelah tombol Next pada tampilan lokasi instalasi dipilih. Pada tampilan ini, nama folder program dapat ditentukan sendiri atau mengikuti nama default yang diberikan pada saat instalasi. Tampilan folder program dapat dilihat pada Gambar A-6. Proses instalasi dilanjutkan dengan memilih tombol Next.

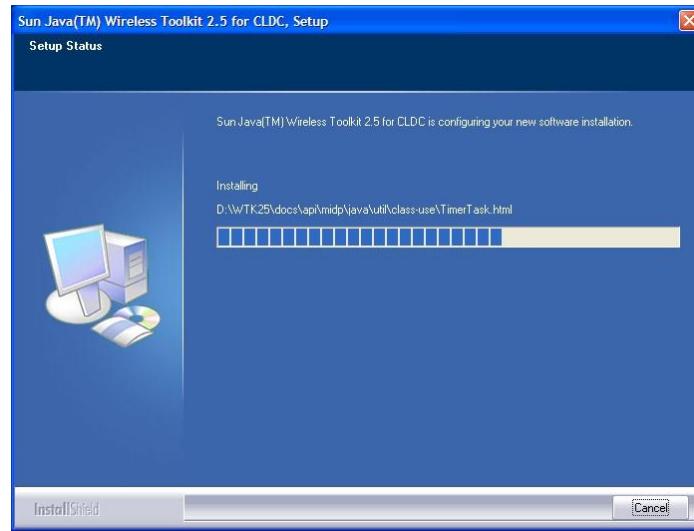
6. Sebelum program instalasi melakukan proses instalasi dan konfigurasi Sun Java Wireless Toolkit ke sistem komputer, tampilan setting instalasi yang dilakukan pada proses-proses sebelumnya diperlihatkan, seperti yang diilustrasikan pada Gambar A-7. Jika setting yang dilakukan telah sesuai, pemilihan tombol Next akan memulai proses instalasi dan konfigurasi Sun Java Wireless Toolkit ke sistem komputer. Kedua proses ini diperlihatkan pada Gambar 1.x dan 1.x.



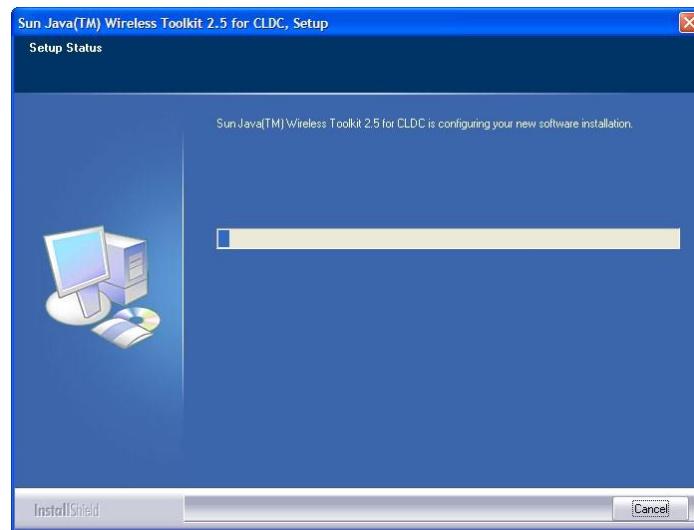
**Gambar A-6** Tampilan Pemilihan Nama Folder Program



**Gambar A-7** Tampilan Setting Instalasi Sun Java Wireless Toolkit

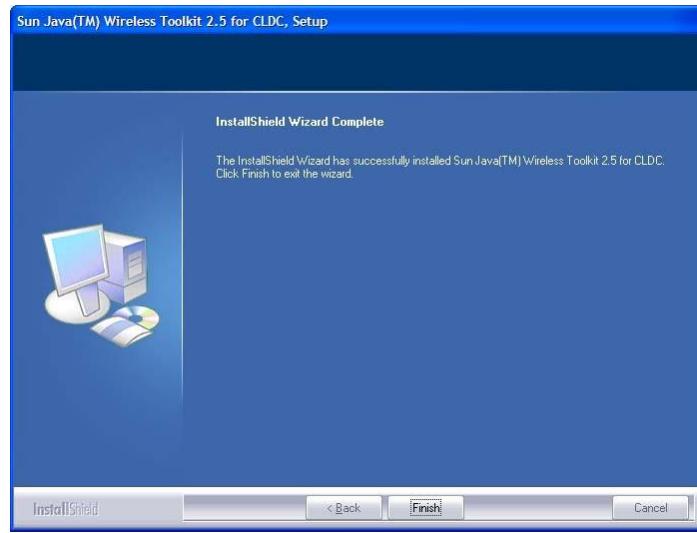


**Gambar A-8** Tampilan Instalasi Sun Java Wireless Toolkit



**Gambar A-9** Tampilan Konfigurasi Sun Java Wireless Toolkit

7. Tampilan yang terakhir memuat informasi apakah proses instalasi berhasil dilakukan. Pada gambar berikut, diinformasikan bahwa Sun Java Wireless Toolkit berhasil diinstalasi ke sistem komputer.



**Gambar A-10** Tampilan Akhir Instalasi Sun Java Wireless Toolkit