

Bab 10

Optional Packages

10.1 Tujuan

Bab ini akan mempelajari tentang penulisan, build, menggunakan emulator dan packaging aplikasi J2ME. IDE yang digunakan adalah NetBeans (www.netbeans.org) dan NetBeans mobility pack.

Setelah menyelesaikan pembahasan bab ini, siswa diharapkan :

- mengetahui fungsionalitas yang disediakan oleh Mobile Media API (MMAPI)
- memainkan nada sederhana
- menjalankan file audio dari jaringan dan file JAR
- mengirim dan menerima pesan SMS
- berkomunikasi wireless menggunakan protokol bluetooth

10.2 Pengenalan

Tidak seluruh device terbuat sama dan tiap class device memiliki fitur yang berbeda – beda pula. Sangatlah sulit untuk membuat spesifikasi standar yang meliputi seluruh device yang telah ada.

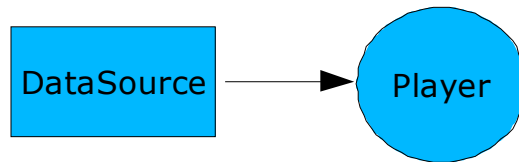
Untuk mengakomodasi perbedaan kemampuan dari device, MIDP memiliki beberapa optional packages. Packages – packages tersebut adalah spesifik dan memenuhi fitur – fitur umum spesifik.

Bab ini akan membahas bagaimana memulai penggunaan Mobile Media API (MMAPI) dan Wireless Messaging API (WMA).

10.3 Mobile Media API (MMAPI)

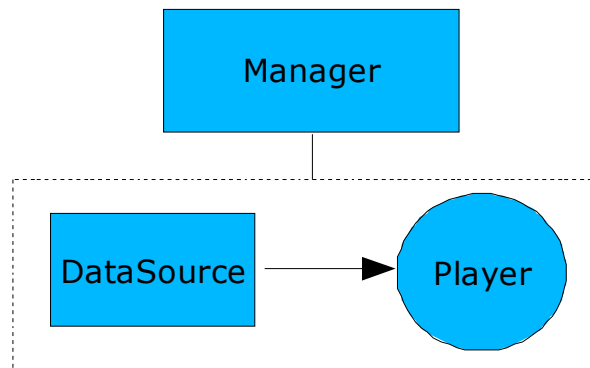
Mobile Media API (MMAPI) memfasilitasi pembuatan nada, memainkan serta merekam audio dan video pada device yang cocok.

Memainkan atau merekam sebuah media ditangani oleh dua object : DataSource dan Player.



DataSource menangani detail cara mendapatkan data dari source yang tersedia. Source dapat berasal dari file JAR atau jaringan (melalui protokol HTTP), record dari RMS, streaming connection dari sebuah server atau sumber proprietary lain. Player tidak perlu terlalu mempermasalahkan darimana data berasal atau bagaimana cara mendapatkannya. Player hanya perlu membaca data yang berasal dari DataSource, memproses, menampilkan dan memainkan playback media pada output device.

Pihak ketiga dalam skenario ini adalah Manager. Manager membuat Player dari DataSource. Manager memiliki method untuk membuat Player dari lokasi sumber media (URL), DataSource dan InputStreams.



Anda dapat menjalankan query terhadap properties MMAPI melalui **String System.getProperty(String key)**.

Key	Deskripsi
microedition.media.version	Versi dari spesifikasi MMAPI yang diterapkan oleh device. Contoh : "1.1"
supports.mixing	Menghasilkan return value "true" jika device mendukung audio mixing : dapat memainkan minimal dua nada secara bersamaan, dapat memiliki minimal dua

Key	Deskripsi
	player yang memainkan audio secara simultan, serta dapat memainkan sebuah nada meskipun paling tidak satu Player memainkan audio pada waktu yang sama
supports.audio.capture	Menghasilkan return value "true" jika mendukung fitur audio capture, dan juga sebaliknya akan dihasilkan value "false"
supports.video.capture	Menghasilkan return value "true" jika mendukung fitur video capture, dan juga sebaliknya akan dihasilkan value "false"
supports.recording	Menghasilkan return value "true" jika mendukung fitur perekaman.

10.3.1 Pembuatan Nada

Memainkan sebuah nada cukup dilakukan dengan memanggil static method `Manager.playTone(int tone, int duration, int volume)`. Nilai yang valid untuk nada adalah antara 0 hingga 127. Durasi dalam memainkan nada diatur dalam ukuran millisecond. Parameter volume memiliki jangkauan antara 0 hingga 100.

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

import java.io.*;

public class ToneMIDlet extends MIDlet implements CommandListener{
    private Command exitCommand, playCommand;
    private Form form;
    private Gauge volumeGauge;
    private Gauge durationGauge;
    private Gauge toneGauge;
    private Display display;
    private int duration = 2; // seconds
    private int volume = 100;
    private int tone = ToneControl.C4;
    private static int MAX_VOLUME = 100;
    private static int MAX_TONE = 127;
    private static int MAX_DURATION = 5;
    public ToneMIDlet() {
        playCommand = new Command("Play", Command.OK, 1);
        exitCommand = new Command("Exit", Command.EXIT, 1);
        volumeGauge = new Gauge("Volume", true, MAX_VOLUME, volume);
        toneGauge = new Gauge("Tone", true, MAX_TONE, tone);
        durationGauge = new Gauge("Duration", true, MAX_DURATION, duration);

        form = new Form("Tone Player");
        form.addCommand(playCommand);
    }
}
```

```
        form.addCommand(exitCommand);
        form.append(volumeGauge);
        form.append(durationGauge);
        form.append(toneGauge);
    }

    public void startApp() {
        display = Display.getDisplay(this);
        form.setCommandListener(this);
        display.setCurrent(form);
    }

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

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            notifyDestroyed();
        }
        if (c == playCommand){
            try {
                volume = volumeGauge.getValue();
                tone = toneGauge.getValue();
                duration = durationGauge.getValue();
                Manager.playTone(tone, duration*1000, volume);
            } catch (MediaException mex){}
        }
    }
}
```

10.3.2 Audio Playback

Method `Manager.createPlayer(String URI)` memudahkan pembuatan sebuah `Player` yang akan memainkan data dari `URI`.

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

import java.io.*;

public class NetAudioMidlet extends MIDlet implements CommandListener{
    private Command exitCommand, playCommand;
    private Form form;
    private Gauge volumeGauge;
    private Display display;
    private int volume = 100;
    private static int MAX_VOLUME = 100;
    Player player;

    public NetAudioMidlet() {
        playCommand = new Command("Play", Command.OK, 1);
        exitCommand = new Command("Exit", Command.EXIT, 1);
        volumeGauge = new Gauge("Volume", true, MAX_VOLUME, volume);
        form = new Form("Audio Player");
    }
}
```

```
        form.addCommand(playCommand);
        form.addCommand(exitCommand);
        form.append(volumeGauge);
    }

    public void startApp() {
        display = Display.getDisplay(this);
        form.setCommandListener(this);
        display.setCurrent(form);
        try {
            player = Manager.createPlayer(
                "http://localhost:8084/Chapter07/bong.wa
                v");

            player.realize();
            // pre-fetch media untuk mengurangi latency
            player.prefetch();
        } catch (IOException ioex) {
            display.setCurrent(new Alert("IO Exception",
                ioex.getMessage(),
                null, AlertType.ERROR));
        } catch (MediaException mex) {
            display.setCurrent(new Alert("Media Exception",
                mex.getMessage(),
                null, AlertType.ERROR));
        }
    }

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

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            notifyDestroyed();
        }
        if (c == playCommand){
            try {
                VolumeControl control = (VolumeControl)
                    player.getControl("VolumeControl");
                if (control != null){
                    control.setLevel(volumeGauge.getValue());
                }

                player.start();
            } catch (MediaException mex) {
                display.setCurrent(new Alert("Media Exception",
                    mex.getMessage(), null, AlertType.ERROR));
            } catch (Exception ex){
                display.setCurrent(new Alert("Exception",
                    ex.getMessage(), null, AlertType.ERROR));
            }
        }
    }
}
```

Anda juga dapat memainkan media yang berasal dari file JAR dengan membuat Stream dari resource file dan meneruskannya pada method `Manager.createPlayer()`

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

import java.io.*;

public class AudioMidlet extends MIDlet implements CommandListener{
    private Command exitCommand, playCommand;
    private Form form;
    private Gauge volumeGauge;
    private Display display;
    private int volume = 100;
    private static int MAX_VOLUME = 100;
    Player player;
    public AudioMidlet() {
        playCommand = new Command("Play", Command.OK, 1);
        exitCommand = new Command("Exit", Command.EXIT, 1);
        volumeGauge = new Gauge("Volume", true, MAX_VOLUME, volume);
        form = new Form("Audio Player");
        form.addCommand(playCommand);
        form.addCommand(exitCommand);
        form.append(volumeGauge);
    }
    public void startApp() {
        display = Display.getDisplay(this);
        form.setCommandListener(this);
        display.setCurrent(form);
    }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}
    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            notifyDestroyed();
        }
        if (c == playCommand){
            try {
                InputStream stream = getClass().
                    getResourceAsStream("bong.wav");
                player = Manager.createPlayer(stream, "audio/x-wav");
                player.realize();
                VolumeControl control = (VolumeControl)
                    player.getControl("VolumeControl");
                if (control != null){
                    control.setLevel(volumeGauge.getValue());
                }
                player.start();
            } catch (MediaException mex) {
                display.setCurrent(new Alert("Media Exception",
                    mex.getMessage(), null, AlertType.ERROR));
            } catch (Exception ex){
                display.setCurrent(new Alert("Exception",
                    ex.getMessage(), null, AlertType.ERROR));
            }
        }
    }
}
```

```
    }  
  }  
}
```

10.4 Wireless Messaging API (WMA)

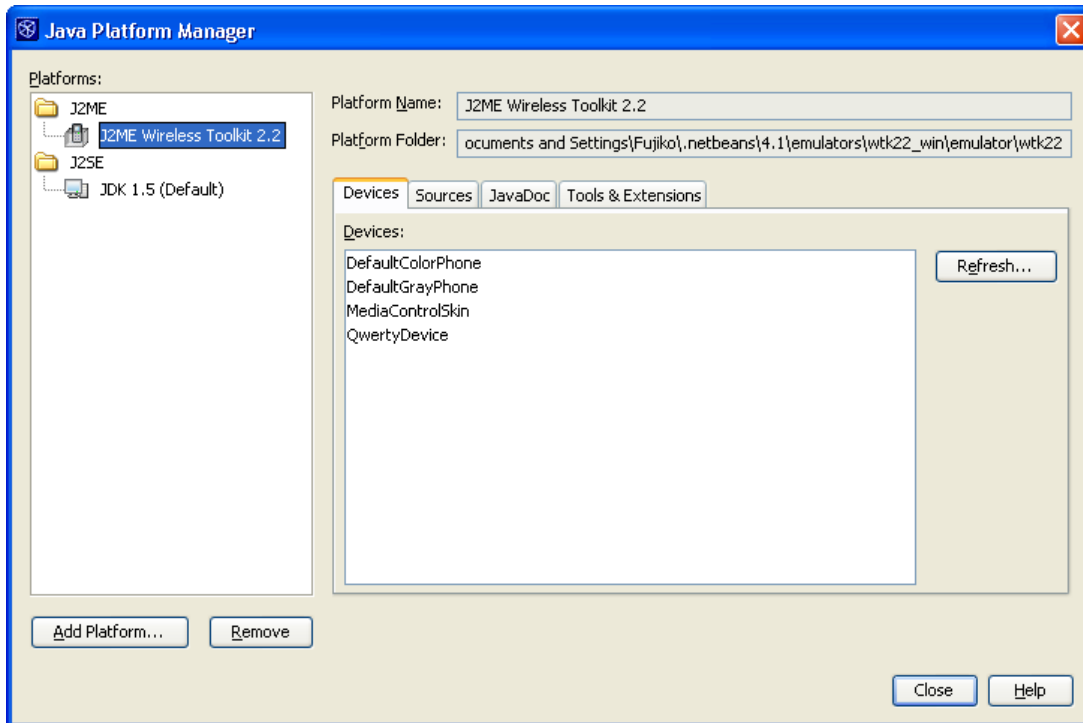
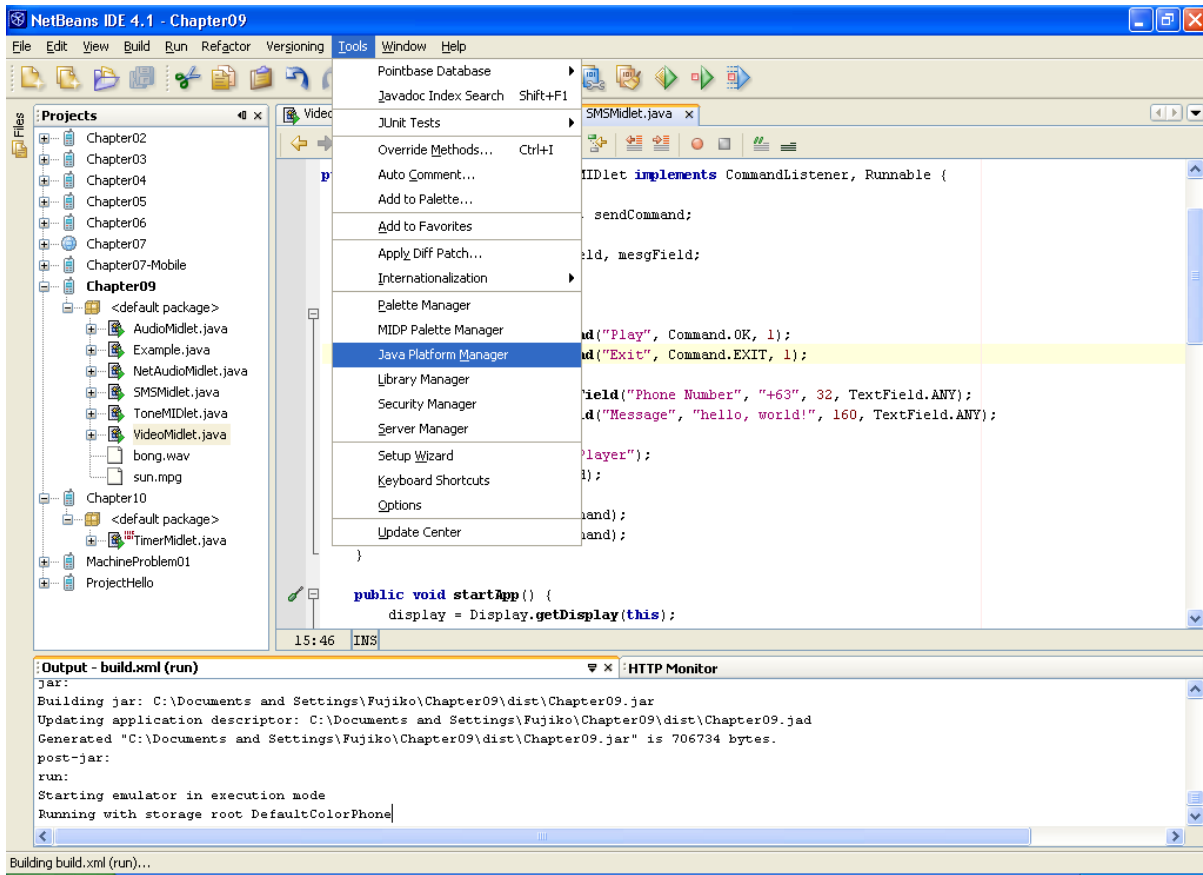
10.4.1 Mengirim SMS

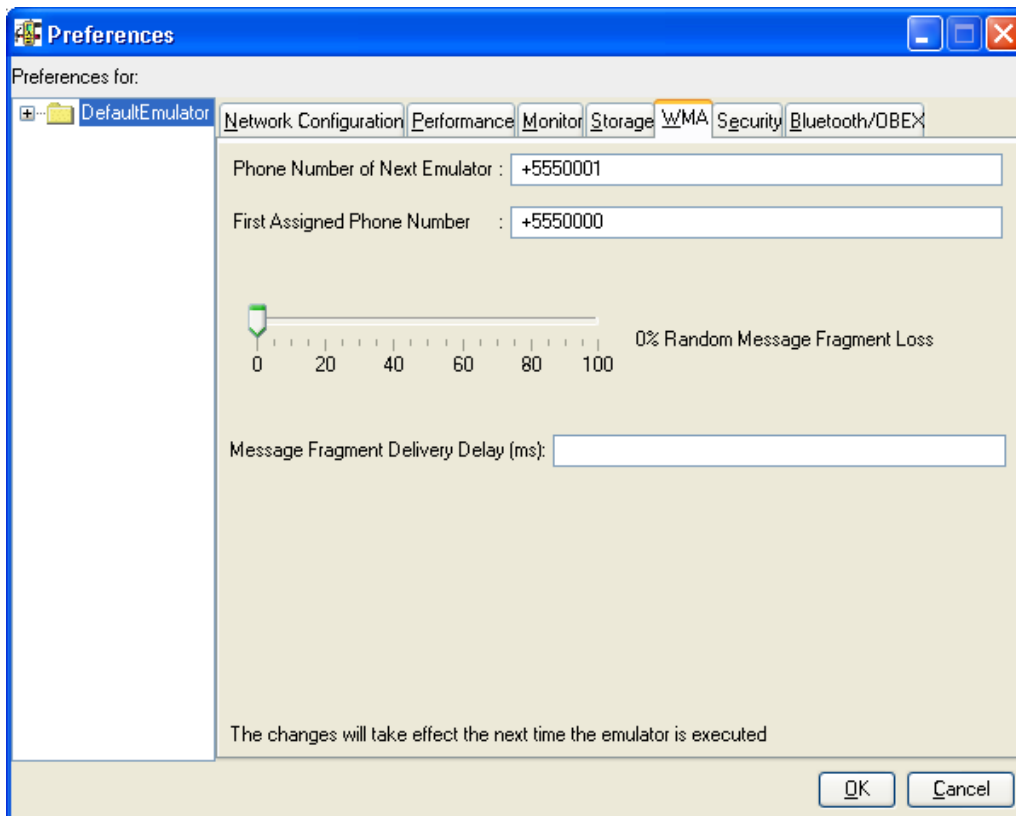
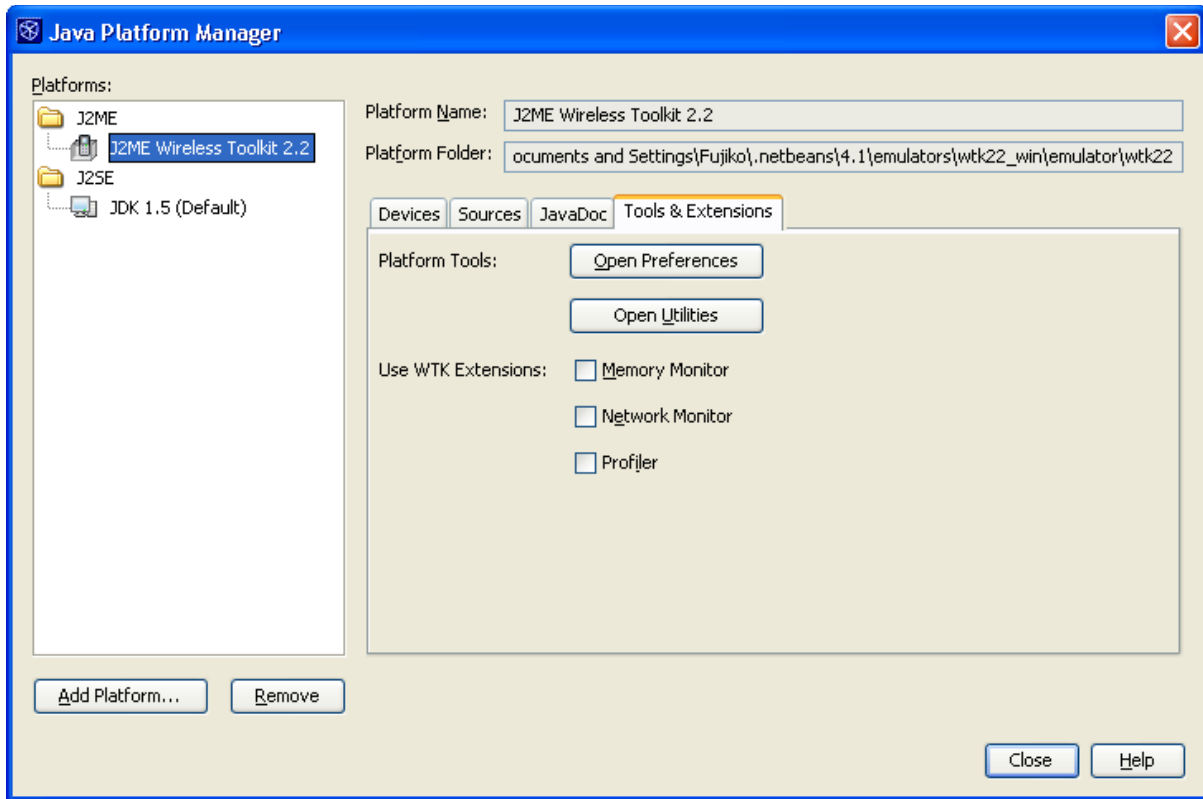
Menggunakan Wireless Messaging API serupa dengan ketika menyambungkan via Socket dan Datagram. Dalam kenyataannya, hal tersebut menggunakan framework yang sama - the Generic Connection Framework (GCF). Format koneksi URL yang digunakan adalah "sms://+639178888888", dimana "+639178888888" adalah nomor telepon yang ingin Anda kirimkan sebuah pesan.

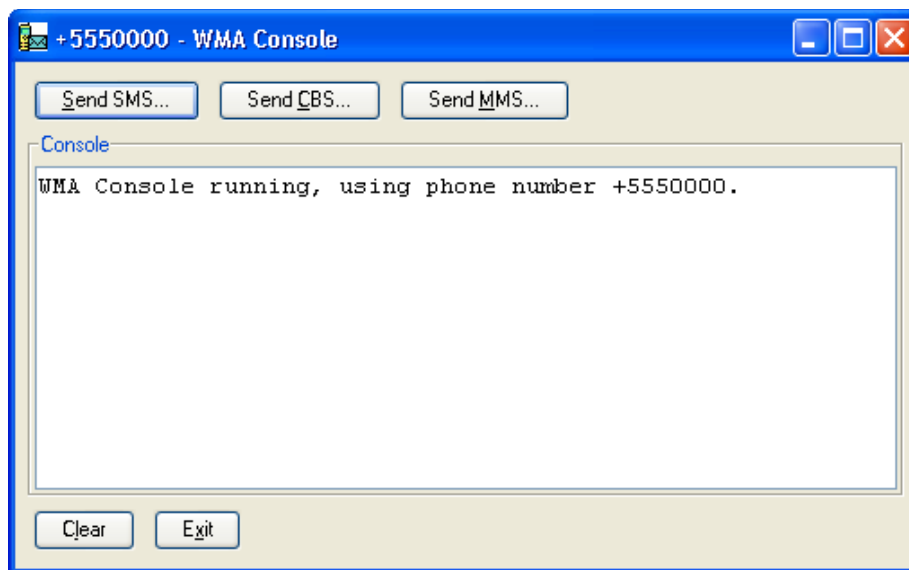
```
public void sendSMS(String number, String message) throws Exception{  
    String url = "sms://" + number;  
  
    MessageConnection connection =  
  
        (MessageConnection) Connector.open(url);  
  
    TextMessage msg = (TextMessage) connection.newMessage(  
        MessageConnection.TEXT_MESSAGE);  
  
    msg.setPayloadText(message);  
  
    connection.send(msg);  
  
    connection.close();  
  
}
```

Pengembangan aplikasi wireless pada NetBeans Mobility Pack 4.1 sangat nyaman. Anda tidak perlu untuk mencoba mengirim pesan SMS hanya untuk mengetes aplikasi Anda. NetBeans Mobility Pack hadir dengan J2ME Wireless Toolkit. Toolkit ini hadir dengan emulator di dalamnya. Dan juga terdapat tool-tool untuk tes mengirim dan menerima pesan SMS. Anda dapat mengkonfigurasi nomor telepon dari telepon yang di-emulasikan dengan membuka pilihan pada WMA.

- Tools
- Java Platform Manager
- J2ME Wireless Toolkit 2.2
- Tool dan Ekstensi :
 1. Membuka Preferences -> WMA
 2. Membuka Utility -> WMA: Open Console







```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import javax.wireless.messaging.*;

public class SMSMidlet extends MIDlet implements CommandListener,
Runnable {

    private Command exitCommand, sendCommand;
    private Form form;
    private TextField addressField, mesgField;
    private Display display;

    public SMSMidlet() {
        sendCommand = new Command("Send", Command.OK, 1);
        exitCommand = new Command("Exit", Command.EXIT, 1);

        addressField = new TextField(
            "Phone Number", "+5550000", 32, TextField.ANY);
        mesgField = new TextField(
            "Message", "hello, world!", 160, TextField.ANY);

        form = new Form("SMS Message");
        form.append(addressField);
        form.append(mesgField);
        form.addCommand(sendCommand);
        form.addCommand(exitCommand);
    }

    public void startApp() {
        display = Display.getDisplay(this);
        form.setCommandListener(this);
        display.setCurrent(form);
    }

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

```
public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) {
        notifyDestroyed();
    }
    if (c == sendCommand) {
        Thread thread = new Thread( this );
        thread.start();
    }
}

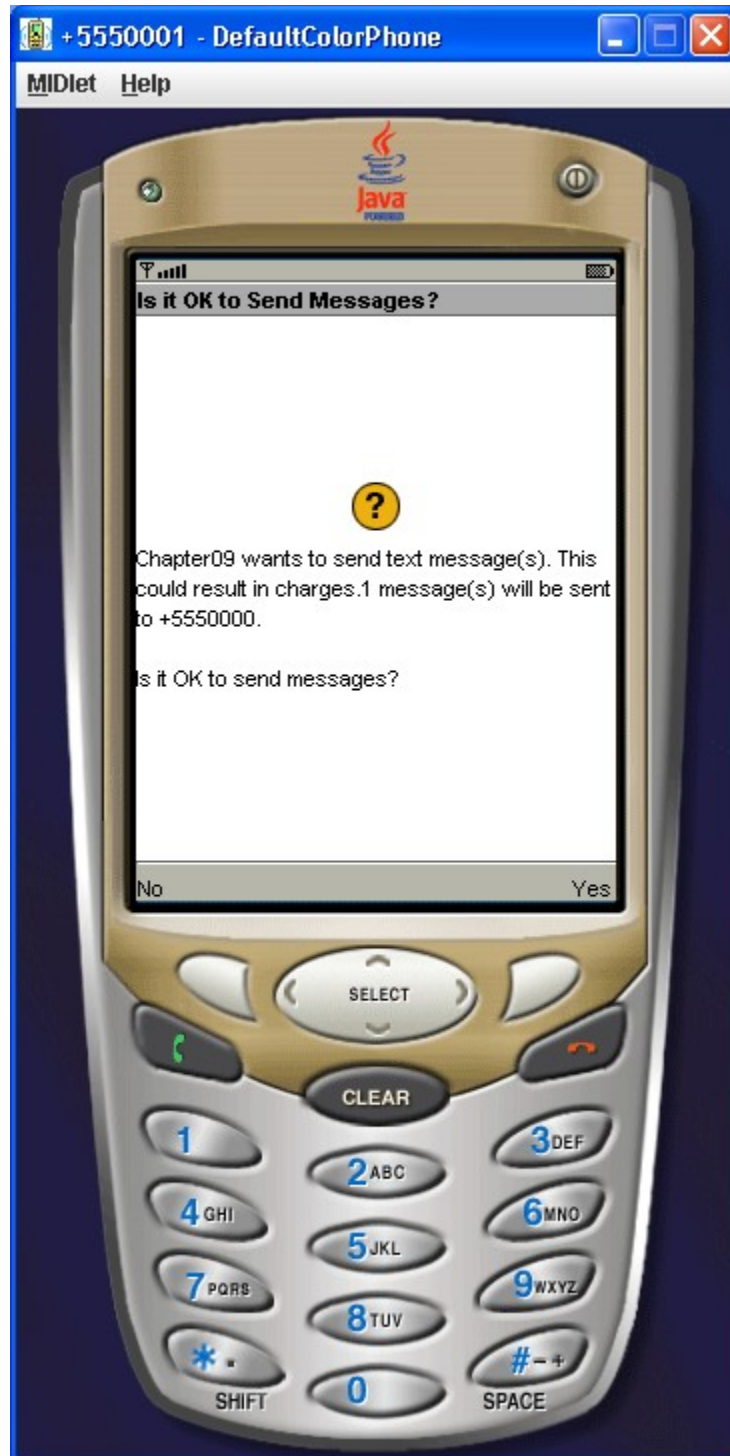
/**
 * Sends an SMS message to number. This method will throw an
exception
 * if there is an error in connecting or sending the message.
 * @param number
 * @param message
 */
public void sendSMS(String number, String message) throws Exception{
    String url = "sms://" + number;
    MessageConnection connection =
        (MessageConnection) Connector.open(url);
    TextMessage msg = (TextMessage) connection.newMessage(
        MessageConnection.TEXT_MESSAGE);
    msg.setPayloadText(message);
    connection.send(msg);
    connection.close();
}

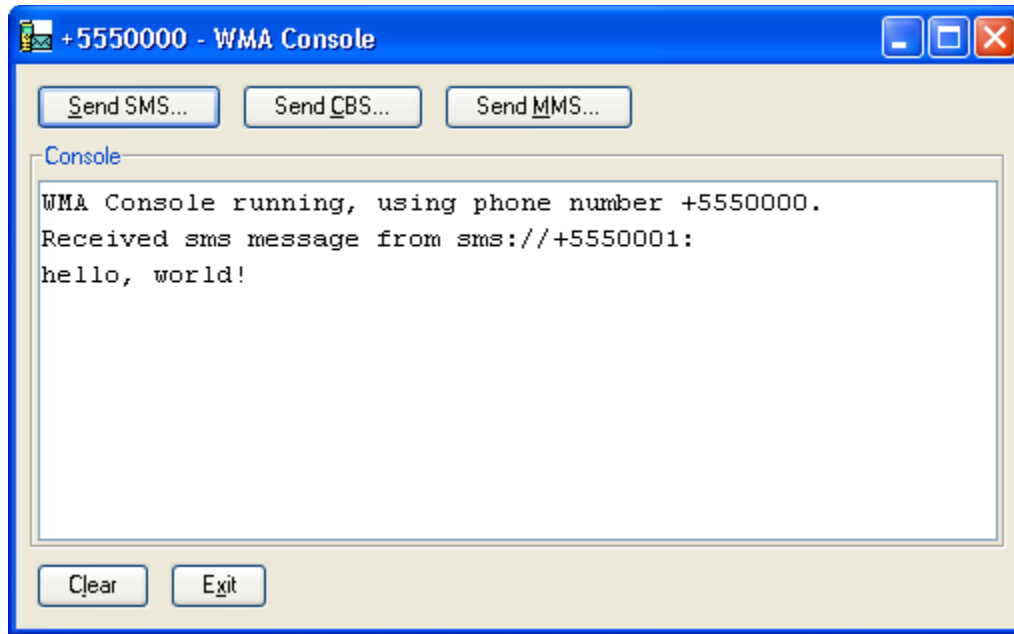
public void run() {
    try {
        String address = addressField.getString();
        String message = mesgField.getString();
        sendSMS(address, message);
        display.setCurrent(new Alert("SMS Message",
            "Message Sent\n"
            + "To: " + address + "\n"
            + "Message: " + message,
            null, AlertType.INFO));
    }
}
```

```
    } catch (Exception ex) {  
        display.setCurrent(new Alert("SMS Error", ex.getMessage(),  
            null, AlertType.ERROR));  
    }  
}  
}
```









10.4.2 Menerima SMS

Untuk menerima sebuah pesan teks, buka sebuah port yang spesifik dari MessageConnection. String protocol untuk pesan SMS adalah "sms". Perintah ini akan menangkap kedatangan pesan SMS dari port 8888:

```
conn = (MessageConnection) Connector.open("sms://:8888");
```

Kita harus mendaftarkan aplikasi Kita untuk menjadi sebuah Message Listener sehingga AMS akan memperhatikan MIDlet Kita dari pesan yang masuk.

```
conn.setMessageListener(this);
```

NotifyIncomingMessage akan dipanggil oleh AMS ketika sebuah sebuah pesan diterima oleh perangkat. Kita akan membutuhkan pembuatan sebuah Thread yang terpisah untuk membaca pesan sehingga method panggilan ulang Listener dapat berakhir tiba-tiba.

```
public void notifyIncomingMessage(MessageConnection messageConnection) {
    if (thread == null){
        thread = new Thread(this);
        thread.start();
    }
}
```

```
    }  
}
```

Dalam method run() Kita, Kita telah siap untuk mendapatkan sebuah pesan :

```
public void run(){  
    try {  
        // menunggu dan menerima pesan  
        Message mesg = conn.receive();  
  
        // menerima pesan  
        // Mengecek apakah pesan berbentuk Teks  
        if (mesg != null && mesg instanceof TextMessage) {  
            TextMessage text = (TextMessage) mesg;  
            addressField.setText(text.getAddress());  
            mesgField.setText(text.getPayloadText());  
            dateField.setText("" + text.getTimestamp());  
            statusField.setText("Message received.");  
        }  
    } catch (Exception e) {  
        statusField.setText("Error: " + e.getMessage());  
    }  
    thread = null;  
}
```

Berikut adalah listing program yang lengkap untuk penerima SMS :

```
import javax.microedition.midlet.*;  
import javax.microedition.lcdui.*;  
import javax.microedition.io.*;  
import javax.wireless.messaging.*;  
  
public class SMSReceiverMidlet extends MIDlet  
    implements CommandListener, MessageListener, Runnable {  
  
    private Command exitCommand, sendCommand;  
    private Form form;  
    private StringItem statusField, addressField, mesgField, dateField;  
    private Display display;
```

```
private MessageConnection conn;
private Thread thread;
private String port = "8888";

public SMSReceiverMidlet() {
    exitCommand = new Command("Exit", Command.EXIT, 1);

    statusField = new StringItem("Status:", "");
    addressField = new StringItem("From:", "");
    mesgField = new StringItem("Message:", "");
    dateField = new StringItem("Timestamp:", "");

    form = new Form("SMS Receiver");
    form.append(statusField);
    form.append(addressField);
    form.append(mesgField);
    form.append(dateField);
    form.addCommand(exitCommand);
}

public void startApp() {
    display = Display.getDisplay(this);
    form.setCommandListener(this);

    startReceiver();
    display.setCurrent(form);
}

public void pauseApp() {
    thread = null;
}

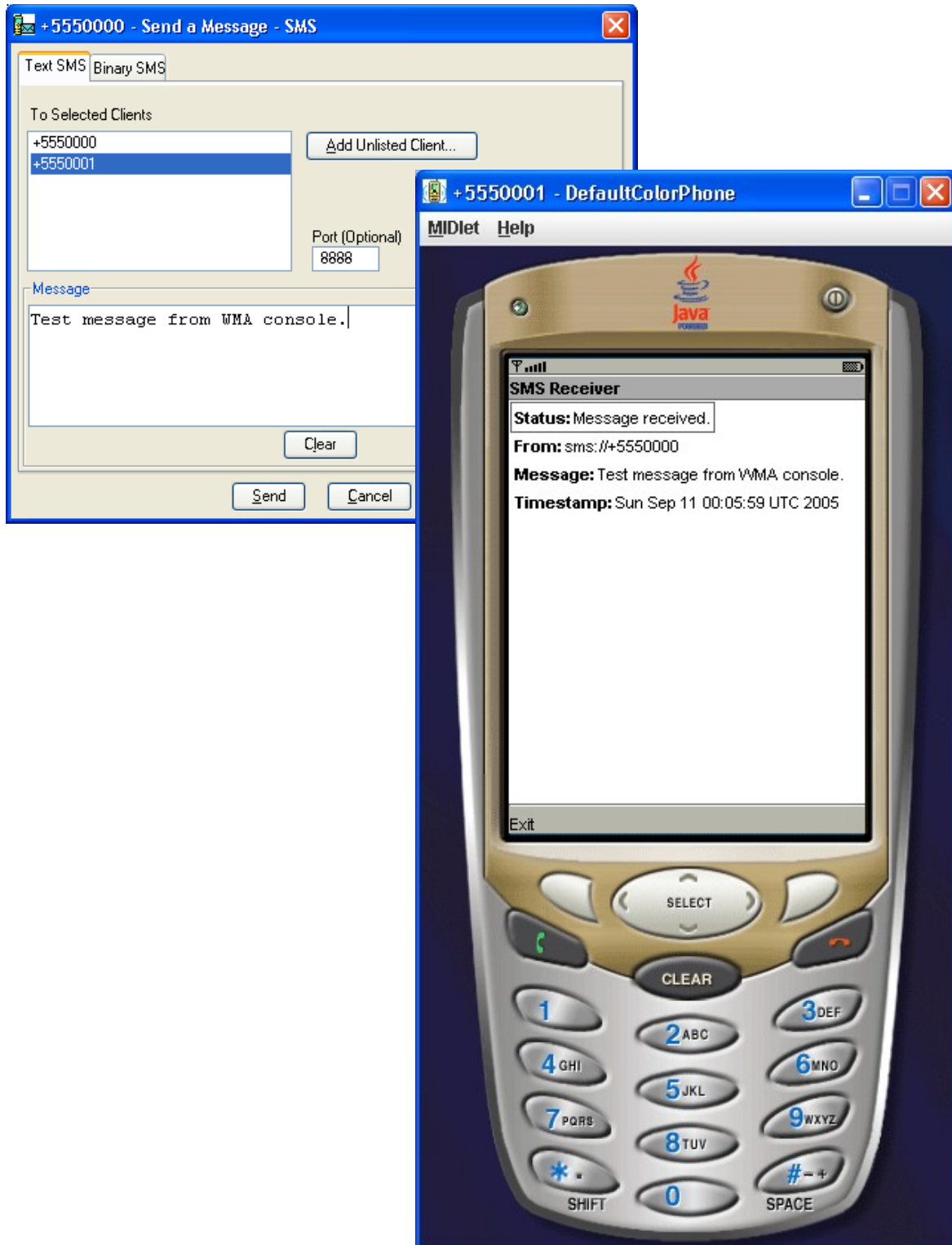
public void destroyApp(boolean unconditional) {
    thread = null;
    if (conn != null){
        try {
            conn.close();
        } catch (Exception ex){}
    }
}
```

```
    }  
}  
  
public void commandAction(Command c, Displayable d) {  
    if (c == exitCommand) {  
        notifyDestroyed();  
    }  
}  
  
private void startReceiver(){  
    try {  
        String addr = "sms://:" + port;  
        if (conn == null){  
            conn = (MessageConnection) Connector.open(addr);  
  
            conn.setMessageListener(this);  
            statusField.setText(  
                "waiting for message at port " + port);  
        }  
    } catch (Exception ex){  
        statusField.setText("Cannot open connection on port "  
            + port + ":" + ex.getMessage());  
    }  
  
    thread = new Thread(this);  
    thread.start();  
}  
  
public void notifyIncomingMessage(MessageConnection messageConn){  
    if (thread == null){  
        thread = new Thread(this);  
        thread.start();  
    }  
}  
  
public void run(){  
    try {
```

```
// menunggu dan menerima pesan
Message mesg = conn.receive();

// menerima pesan
// mengecek apakah pesan berbasis teks
if (mesg != null && mesg instanceof TextMessage) {
    TextMessage text = (TextMessage) mesg;
    addressField.setText(text.getAddress());
    msgField.setText(text.getPayloadText());
    dateField.setText("" + text.getTimestamp());
    statusField.setText("Message received.");
} else {
    statusField.setText(
        "Non-text message received: "
        + mesg.getClass().toString());
}
} catch (Exception e) {
    statusField.setText("Error: " + e.getMessage());
}
thread = null;
}
}
```





10.5 Penggunaan Bluetooth bagi komunikasi Wireless

Gambaran Umum

Bluetooth telah didesain bagi alat komunikasi personal yang mendukung komunikasi tanpa kabel(wireless) seperti pada mobile phone maupun PDA dengan jarak sampai 10 kilometer. Bluetooth adalah sebuah protokol komunikasi yang beroperasi pada frekuensi 2.4 GHz. Sinyal dari bluetooth adalah omni-directional yang dapat menembus tembok. Ia dapat menerima data dan juga suara. Device yang mendukung komunikasi bluetooth dapat dengan mudah menemukan koneksi dan juga berkomunikasi dengan device yang lain secara otomatis.

Bluetooth dapat digunakan untuk mengirim file, membangun sebuah jaringan tertentu, sinkronisasi data, mengkoneksikan sekitar misalnya dengan hands-free kits dan gaming.

Berikut ini adalah beberapa karakteristik dari Bluetooth:

1. Tiga penggolongan yang berbeda berdasarkan jangkauan: kelas 1: 100m, kelas2 = 20m, dan kelas 3=10m
2. Kecepatan 1Mb/s
3. Mengonsumsi energi yang cukup rendah
4. Dapat mentransfer baik suara maupun data
5. Menggunakan signal Omni-direction
6. 2.4GHz – 2.482GHz band

Jaringan Bluetooth

Bluetooth devices dibagi lagi menjadi group-group kecil yang disebut piconet. Didalam piconet, ada sebuah master dan satu atau lebih slaves. Sampai 7 buah slaves bisa diterima didalam sebuah piconet. Sedangkan master unit adalah sesuatu yang memulai proses komunikasi. Ia akan menggunakan komunikasi point-to-multipoint.

Sebuah device didalam sebuah piconet mampu berkomunikasi dengan bluetooth device didalam piconet yang lain. Sebuah slave didalam piconet tertentu mungkin dapat menjadi slave didalam piconet yang berbeda. Sebuah master didalam sebuah piconet dapat menjadi slave pada piconet yang lain. Komunikasi antar piconet tersebut dapat berasal dari jaringan manapun.

Dalam rangka untuk pemeliharaan baterai, bluetooth memiliki tiga low modus operasi yang hemat energi:

Pada modus sniff, sebuah slave device akan memperhatikan berkurangnya level energi, dimana ia tidak berpengaruh terhadap piconet.

Pada modus hold, sebuah device tidak hanya mengirimkan data tetapi ia juga melakukan sinkronisasi secara konstan dengan master. Ia bukanlah member aktif dari piconet, tetapi ia menyimpan alamat member yang aktif.

Sebuah device pada modus park berlaku seperti device pada modus hold, akan tetapi ia tidak menyimpan alamat member yang aktif.

Profile Bluetooth

Profile dari bluetooth telah ditentukan dengan memperhatikan interoperability antara device dan aplikasi dari beberapa manufaktur. Sebuah profile mendefinisikan roles dan kemampuan yang dimiliki oleh tipe aplikasi tertentu. Sebuah device hanya dapat berkomunikasi dengan device yang lain apabila mereka memiliki profile yang sama.

Semua bluetooth device harus menggunakan Generic Access Profile. Profile ini menentukan prosedur koneksi, device discovery, dan management link.

Sebuah Service Discovery Profile mendefinisikan fitur dan prosedur bagi aplikasi bluetooth untuk mengenali segala servis yang telah ditentukan pada bluetooth device yang lain.

Sebuah profile sinkronisasi menentukan hal-hal apa saja yang dibutuhkan pada dua atau lebih devices untuk mensinkronisasikan data.

Profile bluetooth yang lain telah didefinisikan juga pada spesifikasi bluetooth. Akan tetapi tidak akan dibicarakan dalam materi ini.

Keamanan pada Bluetooth

Spesifikasi dari Bluetooth telah menyediakan spesifikasi untuk keamanan dalam tiga hal. Pertama dengan menggunakan frekuensi, yang diharapkan dapat membuat eavesdropper dari komunikasi bluetooth mengalami kesulitan. Limit pada koneksi autentikasi pada device tertentu. Enkripsi dengan secret key untuk membuat data tidak dapat dibaca oleh eavesdropper.

Generic Access Profile telah mendefinisikan sebuah security model yang melingkupi tiga modus security:

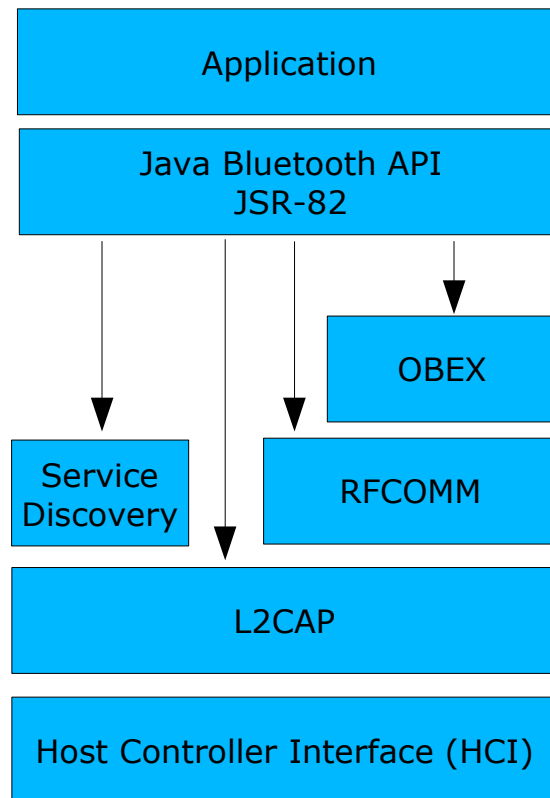
Mode 1: Sebuah modus yang tidak aman karena tidak adanya prosedur keamanan.

Mode 2: Keamanan yang berada pada level servis. Tidak ada prosedur keamanan yang diinisialisasi sebelum channel komunikasi dibangun. Aplikasi mungkin mendapatkan kesulitan policy akses.

Mode 3: Keamanan yang berada pada level link. Sebuah prosedur keamanan akan diinisialisasi sebelum menyelesaikan link-setup.

Protocol Stack pada Bluetooth

Berikut ini adalah ilustrasi dari Bluetooth protokol stack yang pertama.



Bluetooth J2ME optional package seperti telah didefinisikan pada pada JSR 82 akan mendukung Anda untuk mengontrol bluetooth device. Untuk mengirimkan data diantara device yang mendukung bluetooth, Anda mungkin dapat menggunakan satu dari tiga koneksi berikut ini: L2CAP, RFCOMM, dan OBEX.

L2CAP –untuk data packet

RFCOMM – satu layer diatas protokol L2CAP dan digunakan untuk data streaming.

OBEX – digunakan untuk data object

Aplikasi RFCOMM

Contoh aplikasi berikut ini menggunakan protokol serial RFCOMM untuk berkomunikasi dengan device bluetooth. Aplikasi ini adalah tulang punggung dari sebuah information server dan client-nya.

Server pertama kali akan mendaftarkan service-nya:

```
localDevice = LocalDevice.getLocalDevice();
localDevice.setDiscoverable(DiscoveryAgent.GIAC);
notifier = (StreamConnectionNotifier) Connector.open(URL);
```

Kemudian ia akan menunggu koneksi:

```
StreamConnection conn = notifier.acceptAndOpen();
```

Sekali sebuah koneksi diterima, kemudian ia akan membaca command dan menjawab dengan jawaban yang tepat dengan menggunakan method untuk koneksi serial.

Client pertama kali akan mencari device yang tersedia untuk service tersebut:

```
LocalDevice localDevice = LocalDevice.getLocalDevice();
discoveryAgent = localDevice.getDiscoveryAgent();

discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
```

Client akan meng-implement `DiscoveryListener` dan meng-override method yang diperlukan untuk menerima notifikasi dari device. Sekali device ditemukan dan proses pencarian servis sudah selesai dilaksanakan, kemudian command tertentu akan dibutuhkan.

InfoServer.java:

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

public class InfoServer implements Runnable {
    InputStream input;
    OutputStream output;
    StreamConnectionNotifier notifier;
    StreamConnection conn;
    LocalDevice localDevice;
    ServiceRecord serviceRecord;
```

```
public static String SERVICE_NAME = "chat";
public static UUID PORT = new UUID(0x0518);

private boolean isRunning = false;
private static String URL = "btspp://localhost:" + PORT +
    ";name=" + SERVICE_NAME + ";authorize=true";

public InfoServer() {
    isRunning = false;
    Thread thread = new Thread(this);
    thread.start();
}

public void run() {

    if (!isRunning) {
        try {
            conn = null;

            localDevice = LocalDevice.getLocalDevice();
            localDevice.setDiscoverable(DiscoveryAgent.GIAC);

            notifier = (StreamConnectionNotifier)
Connector.open(URL);

        } catch (BluetoothStateException e) {
            System.err.println("Bluetooth Exception: " +
e.getMessage());
        } catch (IOException e) {
            System.err.println("IO Exception: " + e.getMessage());
        }

        isRunning = true;
    }

    while (true) {
```

```
    try {
        System.out.println("Waiting for connection...\n");

        // Menunggu koneksi
        conn = notifier.acceptAndOpen();

        // Membaca command
        String msg = BluetoothMidlet.read(conn);
        System.out.println("Received from Client: " + msg);

        // Mengirim balasan
        msg = "InfoServer: Your command was: " + msg;
        output = conn.openOutputStream();
        output.write(msg.length());
        output.write(msg.getBytes());
        output.close();
    } catch (Exception ex) {
        System.err.println("Bluetooth Server Exception: " + ex);
    }
}
}
```

InfoClient.java:

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

class InfoClient implements DiscoveryListener {
    private DiscoveryAgent discoveryAgent;
    private RemoteDevice[] remoteDevices;
    private UUID[] UUIDSet;
    private String URL;

    public InfoClient() {
        try {
```

```
        LocalDevice localDevice = LocalDevice.getLocalDevice();
        discoveryAgent = localDevice.getDiscoveryAgent();

        discoveryAgent.startInquiry(DiscoveryAgent.GIAC, this);
    } catch (Exception e) {
        System.out.println(e);
    }
}

public void deviceDiscovered(RemoteDevice btDevice, DeviceClass
cod) {
    try {
        // Mendapatkan informasi dari device
        System.out.println("deviceDiscovered()");
        System.out.println("Address: " +
btDevice.getBluetoothAddress());
        System.out.println("Major Device Class: " +
cod.getMajorDeviceClass());
        System.out.println("Minor Device Class: " +
cod.getMinorDeviceClass());
        System.out.println("Friendly Name: " +
btDevice.getFriendlyName(true));

        UUIDSet = new UUID[1];
        UUIDSet[0] = InfoServer.PORT;

        int searchID = discoveryAgent.searchServices(null, UUIDSet,
btDevice, this);

    } catch (Exception e) {
        System.out.println("Exception: " + e);
    }
}

public void servicesDiscovered(int transID, ServiceRecord[]
servRecord) {
    System.out.println("servicesDiscovered()");

    for (int i=0; i<servRecord.length; i++) {
```

```
        URL = servRecord[i].getConnectionURL(0, false);
    }
}

public void serviceSearchCompleted(int transID, int responseCode) {
    switch (responseCode) {
        case SERVICE_SEARCH_COMPLETED:
            System.out.println("SERVICE_SEARCH_COMPLETED\n");
            System.out.println("Service URL: " + URL);

            StreamConnection conn = null;
            try {
                String msg = "INFO";
                conn = (StreamConnection)Connector.open(URL);
                OutputStream output = conn.openOutputStream();
                output.write(msg.length());
                output.write(msg.getBytes());
                output.close();

                System.out.println(BluetoothMidlet.read(conn));

            } catch (Exception ex) {
                System.out.println(ex);
            } finally {
                try {
                    conn.close();
                } catch (IOException ioe) {
                    System.out.println("Error Closing connection "
+ ioe);
                }
            }

            break;

        case SERVICE_SEARCH_ERROR:
            System.out.println("SERVICE_SEARCH_ERROR\n");
            break;

        case SERVICE_SEARCH_TERMINATED:
```

```
        System.out.println("SERVICE_SEARCH_TERMINATED");
        break;
    case SERVICE_SEARCH_DEVICE_NOT_REACHABLE:
        System.out.println("SERVICE_SEARCH_DEVICE_NOT_REACHABLE
");
        break;
    case SERVICE_SEARCH_NO_RECORDS:
        System.out.println("SERVICE_SEARCH_NO_RECORDS");
        break;

        default: break;
    }
}

public void inquiryCompleted(int discType) {
    System.out.println("inquiryCompleted()");
}
}
```

BluetoothMidlet.java:

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

public final class BluetoothMidlet extends MIDlet implements
CommandListener {

    private final Command okCmd = new Command("Start", Command.OK, 1);
    private final Command exitCmd = new Command("Exit", Command.EXIT,
1);

    private static final String[] commands = { "Server", "Client" };
    private final List menu = new List("Bluetooth Application",
        List.IMPLICIT, commands, null);
```

```
Display display;
private InfoClient chatClient;
private InfoServer chatServer;

public BluetoothMidlet() {
    menu.addCommand(exitCmd);
    menu.addCommand(okCmd);
    menu.setCommandListener(this);
}

public void startApp() {
    display = Display.getDisplay(this);
    display.setCurrent(menu);
}

protected void destroyApp(boolean unconditional) {
}

protected void pauseApp() {
}

public void commandAction(Command c, Displayable d) {
    if (c == exitCmd) {
        destroyApp(true);
        notifyDestroyed();
        return;
    }

    switch (menu.getSelectedIndex()) {
        case 0: chatServer = new InfoServer(); break;
        case 1: chatClient = new InfoClient(); break;
        default: break;
    };
}
```

```
public final static String read(StreamConnection conn) {

    InputStream is = null;
    byte[] dataBytes = null;
    int len;

    try {
        is = conn.openInputStream();

        len = is.read();
        dataBytes = new byte[len];
        len = 0;

        while (len != dataBytes.length) {
            int readLen = is.read(dataBytes, len, dataBytes.length
- len);

            if (readLen == -1) {
                System.err.println("Error reading data.");
            }
            len += readLen;
        }
    } catch (IOException ex) {
        System.err.println(ex);
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch (IOException ex) { }
        }
    }
    return new String(dataBytes);
}
}
```

10.6 Lokasi API

Location-based Services

Location based services memanfaatkan lokasi dari device untuk menyediakan informasi mengenai lokasi. Pertanyaan utama yang dijawab oleh Location Based Service adalah "Dimana saya?" . Untuk lebih spesifiknya adalah "Dimana letak device?".

Aplikasi yang lebih populer mengenai penginformasian lokasi adalah dengan menggunakan telepon genggam, orang tua dapat melacak dimana anaknya sekarang berada. Atau dengan telepon genggam tersebut dapat menemukan toko atau restoran yang paling dekat, mendapatkan informasi mengenai lalu lintas pada lokasi tertentu dan mendapatkan sebuah arahan menuju tempat tertentu. Penggunaan data mengenai lokasi untuk mendapatkan pengalaman yang lebih pada game dan juga bagi aplikasi-aplikasi social networking adalah beberapa contoh dari banyaknya kemungkinan aplikasi yang dapat memanfaatkan penginformasian lokasi.

Lokasi dari device ditentukan oleh salah satu dari beberapa location service yang telah tersedia seperti Global Positioning System (GPS) atau informasi dari jaringan seluler itu sendiri (seperti site ID dari seluler). GPS memanfaatkan network dari satelit untuk digunakan dan dikontrol oleh Departemen pertahanan US. GPS dapat menjadi tidak efektif, apabila ia didalam bangunan atau diantara tumbuh-tumbuhan yang lebat. Akan tetapi, ia sangat akurat sampai 5-30 meter.

Penginformasian lokasi dari dari seluler site dapat digunakan apabila aplikasi tidak membutuhkan banyak akurasi. Lingkup dari cell site sangat bervariasi, dari beberapa kilometer sampai dengan 20 kilometer.

The Location API for J2ME (JSR 179)

`javax.microedition.location` package memiliki class-class yang digunakan untuk mendapatkan informasi mengenai lokasi sebuah device.

Beberapa method khusus digunakan untuk menentukan terminal dari lokasi device. API tersebut akan mengembalikan informasi-informasi berikut ini:

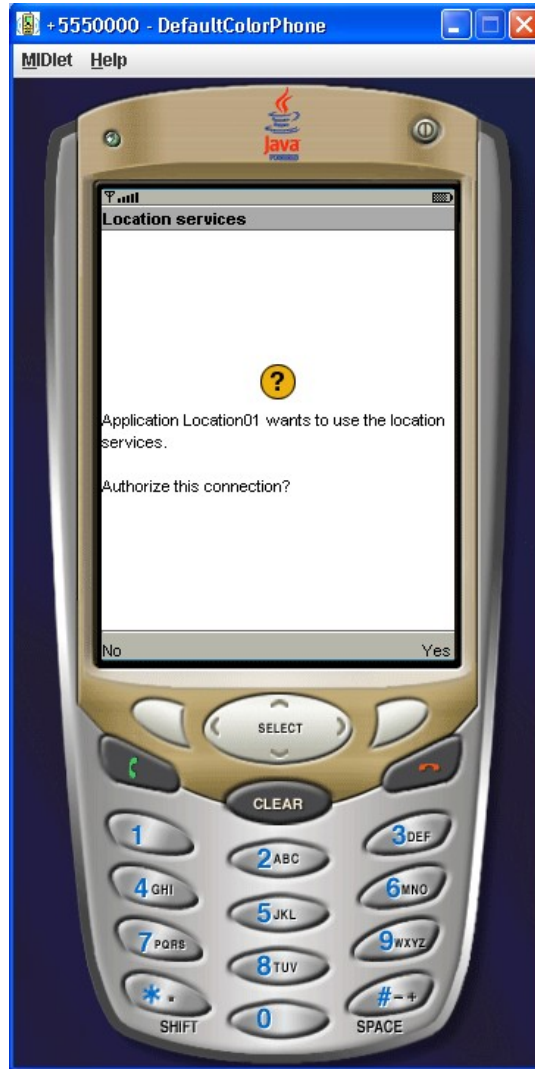
1. Garis Lintang
2. Garis Bujur
3. Keakuratan dari informasi mengenai garis lintang dan bujur
4. Timestamp pada saat penentuan sebuah lokasi

Ketersediaan dari fitur-fitur diatas bergantung pada method yang digunakan untuk menentukan informasi lokasi:

- Ketinggian dari permukaan laut dan keakuratan dari pengukuran tersebut.
- Informasi mengenai kecepatan dan jalan raya
- Informasi mengenai alamat
- Proximity landmark pada event tertentu
- Sebagai kompas yang menunjuk ke arah utara sesuai dengan orientasi dari device
- Informasi mengenai pitch (puncak) dan roll

Location Provider

Location provider berfungsi sebagai sumber dari penginformasian lokasi. Aplikasi dapat menyediakan sebuah kriteria (dimungkinkan juga termasuk akurasi) pada saat mendapatkan instance dari LocationProvider. Method yang mengakses informasi mengenai lokasi akan melalui SecurityException jika aplikasi tidak mendapatkan *permission* untuk mengakses informasi tertentu.



Security

Tabel Loc01 terdiri dari nama-nama permission dan method yang dilindungi oleh permission tersebut.

Nama Permission	Method yang dilindungi
javax.microedition.location.Location	LocationProvider.getLocation(), LocationProvider.setLocationListener()
javax.microedition.location.Orientation	Orientation.getOrientation()
javax.microedition.location.ProximityListener	LocationProvider.addProximityListener()
javax.microedition.location.LandmarkStore	LandmarkStore.getInstance(),

Nama Permission	Method yang dilindungi
ore.read	LandmarkStore.listLandmarkStores()
javax.microedition.location.LandmarkStore.write	LandmarkStore.addLandmark(), LandmarkStore.deleteLandmark(), LandmarkStore.removeLandmarkFromCategory(), LandmarkStore.updateLandmark()
javax.microedition.location.LandmarkStore.category	LandmarkStore.addCategory(), LandmarkStore.deleteCategory()
javax.microedition.location.LandmarkStore.management	LandmarkStore.createLandmarkStore(), LandmarkStore.deleteLandmarkStore()

Landmarks

Location API juga mendukung penyimpanan informasi mengenai landmark dengan menggunakan LandmarkStores. Device yang menggunakan JSR 179 memerlukan paling tidak satu LandmarkStore untuk menyimpan sebuah landmark.

Sebuah landmark adalah sebuah lokasi yang nama-nya diketahui oleh user. Sebuah landmark juga dapat terdiri dari penjelasan-penjelasan (text), garis koordinat, dan informasi mengenai alamat (optional). Sebuah landmark direpresentasikan dengan Location API dalam class **javax.microedition.location.landmark**. Class ini hanya merupakan sebuah container dari informasi.

LandmarkStore class terdiri dari method-method pada proses penyimpanan data, pengambilan kembali data yang sudah ada, dan juga menghapus sebuah landmark didalam sebuah persistent landmark store. Penggunaan dari Location API paling tidak memiliki sebuah landmark store, sebagai default. Implementasi yang lain dari Location API dimungkinkan untuk mendukung lebih dari sebuah landmark store dan kemudian semua data yang disimpan akan digunakan secara bersama-sama oleh beberapa aplikasi.

Penamaan sebuah landmark sangat dibutuhkan dan nama tersebut akan dipresentasikan kepada user. Landmark dapat ditempatkan pada Category nol atau yang lain. Nama sebuah category haruslah unique didalam landmark store, akan tetapi nama sebuah landmark dimungkinkan untuk diduplikasi didalam sebuah landmark store.

Location Listener

Sebuah location listener tunggal dapat mendaftarkan dirinya kepada sebuah instance Landmark provider. Listener ini harus implement LocationListener interface. Method locationUpdate() akan dipanggil berulang kali didalam sistem tergantung kepada interval yang diberikan oleh aplikasi.

Sistem tidak akan menggaransi bahwa method tersebut akan dipanggil pada interval tertentu. Akan tetapi, dimungkinkan location provider memberikan informasi yang tidak akurat terutama apabila location provider tersebut tidak dapat memberikan informasi mengenai lokasi tersebut tepat waktu.

Method `providerStateChanged()` juga akan dipanggil kemudian status dari location provider akan berubah. Implementasi dari method ini akan di-return seketika. Apabila method ini membutuhkan waktu yang cukup lama pada saat implementasi, maka aplikasi tersebut harus menggunakan thread secara terpisah.

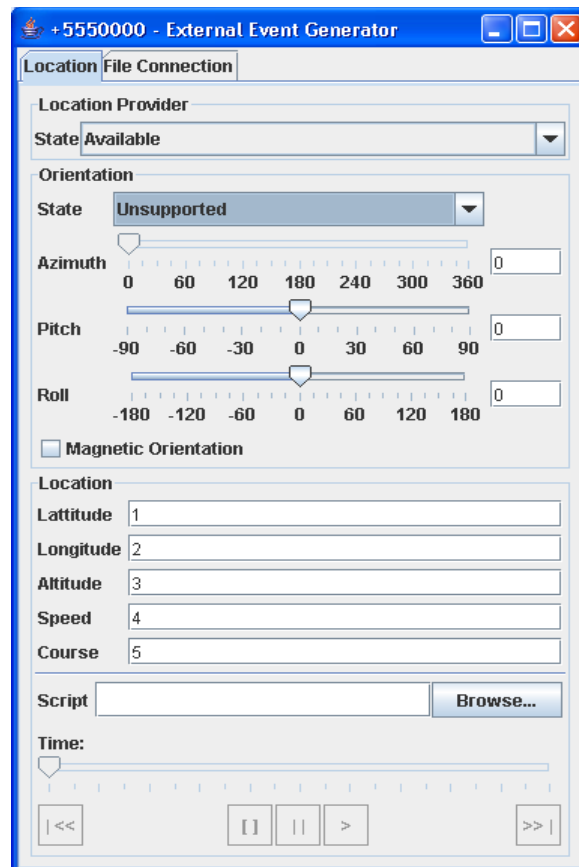
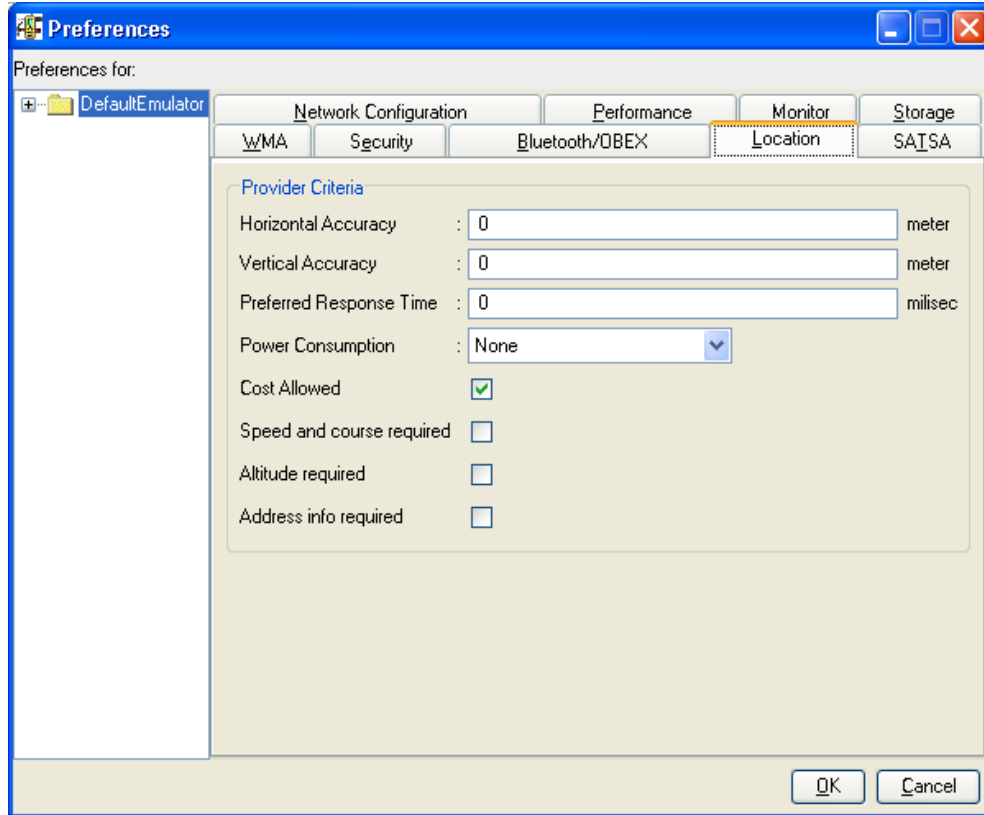
Proximity Listener

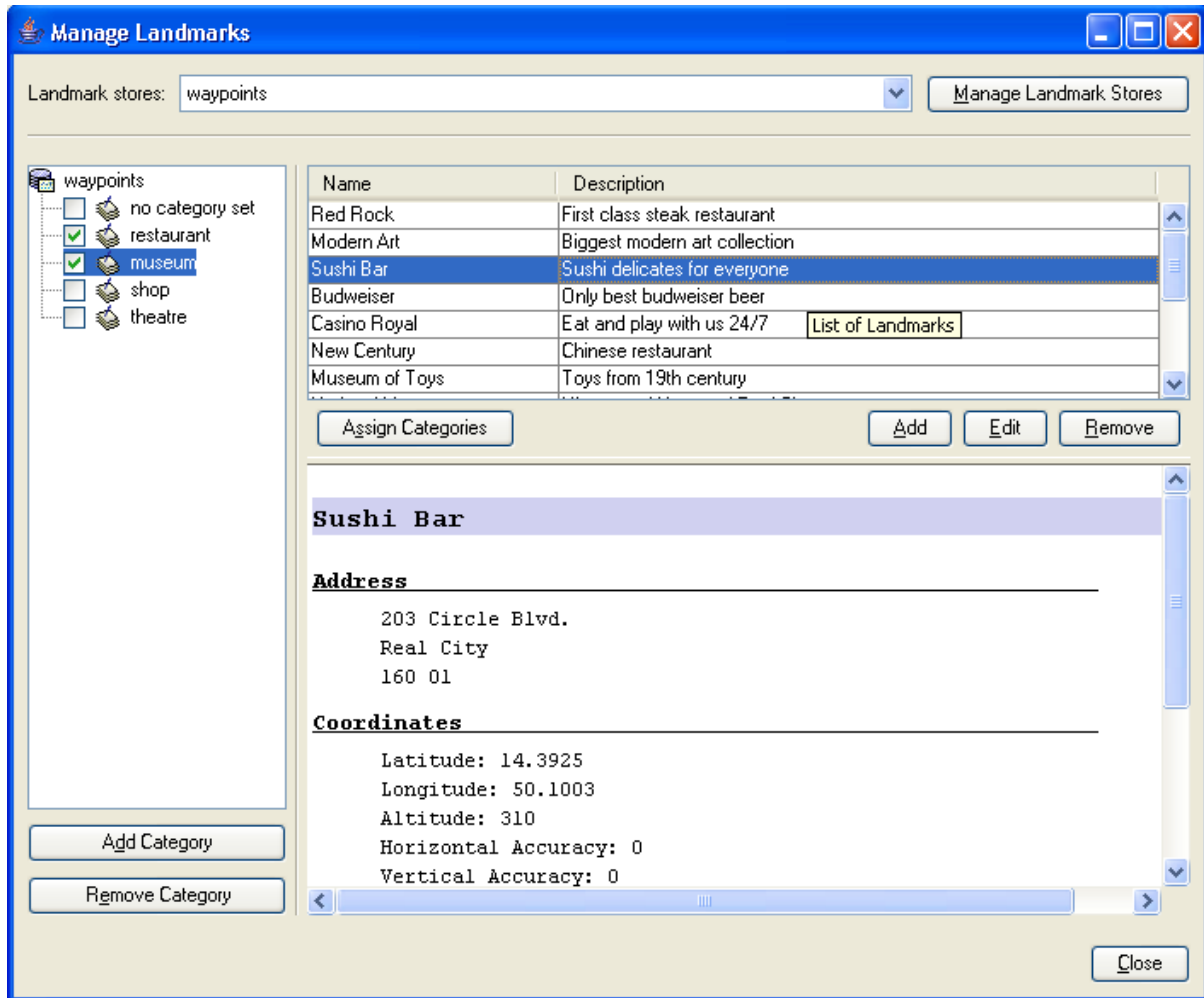
Baik nol maupun lebih proximity listener harus didaftarkan pada object dari landmark provider. Method `proximityEvent()` dari proximity listener akan dipanggil apabila device tersebut terletak didalam jangkauan koordinat yang telah didaftarkan. Sebuah proximity listener harus didaftarkan lagi (jika diperlukan) setelah pemanggilan sebuah method karena proses pendaftaran akan tertunda pada saat listener dipanggil.

Pengidentifikasian Location API

Sebuah aplikasi dapat mengetest eksistensi dan versi dari Location API dengan cara meng-query sistem property dengan kunci "microedition.location.version". Method `System.getProperty` yang telah disediakan dengan kunci tersebut akan me-return versi string. Misalnya, implementasi dari JSR 179 pada versi 1.0 akan me-return string "1.0".

Location API yang didukung oleh Sun Java Wireless Toolkit





Sejak versi 2.3, Sun Java Wireless Toolkit memberikan dukungan pada Location API. Preference pada API dapat dimodifikasi dengan cara: Tools-> Java Platform Manager -> J2ME -> Sun Java Wireless Toolkit -> Tools and Extensions -> Open Preferences -> Location. Landmark dapat di-manage pada bagian Utilities dari Toolkit.

Dari MIDlet window (pada saat Anda menjalankan aplikasi dari emulator), window "External Events" dapat diakses pada menu. Pada tab Location, Anda dapat mensimulasikan dan memodifikasi lokasi device saat ini.

Location01.java:

```

/*
 * LocationMidlet.java
 *
 */

package jedi;

```

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

/**
 *
 * @author jedi
 * @j
 */
public class LocationMidlet extends MIDlet implements CommandListener,
Runnable {
    private final Command exitCmd = new Command("Exit", Command.EXIT,
1);
    private final Command locationCmd = new Command("Location",
Command.ITEM, 1);
    private Display display;
    private Form mainForm = new Form("JEDI: Location Example");
    private StringItem latitude = new StringItem("Latitude:", "",
Item.PLAIN);
    private StringItem longitude = new StringItem("Longitude:", "",
Item.PLAIN);
    private StringItem altitude = new StringItem("Altitude:", "",
Item.PLAIN);
    private StringItem speed = new StringItem("Speed:", "", Item.PLAIN);
    private StringItem course = new StringItem("Course:", "",
Item.PLAIN);
    private StringItem locMethod = new StringItem("Method:", "",
Item.PLAIN);
    private StringItem timestamp = new StringItem("Timestamp:", "",
Item.PLAIN);
    private StringItem status = new StringItem("Status:", "",
Item.PLAIN);
    private StringItem version = new StringItem("Version:", "Unknown",
Item.PLAIN);
    private LocationProvider locationProvider;

    public LocationMidlet() {
        mainForm.addCommand(exitCmd);
        mainForm.addCommand(locationCmd);
        mainForm.setCommandListener(this);
    }
}
```

```
        version.setText(System.getProperty("microedition.location.version"));

        mainForm.append(version);
        mainForm.append(latitude);
        mainForm.append(longtitude);
        mainForm.append(altitude);
        mainForm.append(speed);
        mainForm.append(course);
        mainForm.append(locMethod);
        mainForm.append(timestamp);
        mainForm.append(status);
    }

    public void startApp() {
        display = Display.getDisplay(this);
        display.setCurrent(mainForm);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }

    public void commandAction(Command c, Displayable d) {
        if (c == locationCmd) {
            Thread thread = new Thread(this);
            thread.start();
        }
        if (c == exitCmd) {
            destroyApp(true);
            notifyDestroyed();
            return;
        }
    }

    public void run() {
        try {
```

```
Criteria cr = new Criteria();
// Men-set akurasi Horisontal ke 1 kilometer
cr.setHorizontalAccuracy(0);
cr.setVerticalAccuracy(0);
LocationProvider provider =
LocationProvider.getInstance(cr);

// Timeout setelah 30 detik
Location loc = provider.getLocation(30);

Coordinates coord = loc.getQualifiedCoordinates();
if (coord != null) {
    if (loc.isValid()) {
        latitude.setText(Coordinates.convert(coord.getLatitude(), Coordinates.DD_MM_SS));
        longitude.setText(Coordinates.convert(coord.getLongitude(), Coordinates.DD_MM_SS));
        altitude.setText(Float.toString(coord.getAltitude()));
        timestamp.setText(Long.toString(loc.getTimestamp()));
        speed.setText(Float.toString(loc.getSpeed()));
        locMethod.setText(Integer.toString(loc.getLocationMethod()));
        course.setText(Float.toString(loc.getCourse()));

        status.setText("Valid Location Information");
    } else {
        status.setText("Invalid Location Information");
    }
}

} catch (LocationException e) {
    status.setText("Exception:" + e.getMessage());
} catch (InterruptedException e) {
    status.setText("Timeout: " + e.getMessage());
}
}
```



10.7 Latihan

10.7.1 Audio Player

Buatlah sebuah MIDlet yang dapat memainkan file audio sebagai sebuah indefinite loop. Audio tersebut akan dibaca dari JAR file. Petunjuk: Anda harus menge-set properti dari player untuk mengontrol looping.

10.7.2 SMS Auto-Responder

Buatlah sebuah MIDlet yang secara otomatis akan me-reply apabila ia menerima sebuah text message. Petunjuk: Anda dapat memodifikasi SMSReceiverMidlet dan menggunakan koneksi yang sama untuk me-reply pesan.