Wireless Application Programming with Java

Qusay H. Mahmoud
Dept. of Computing and Information Science
University of Guelph, Canada

qmahmoud@cis.uoguelph.ca
You will learn:

- What is the J2ME platform?
- How does it differ from J2SE?
- KVM (Kilo Virtual Machine)
- Configurations and Profiles
  - CLDC and MIDP
  - CDC, Foundation, Personal, RMI
- How to get started developing wireless applications using the J2ME platform
Outline

- Web Content for Mobile Devices
- WAP Programming Model
- J2ME Platform
- KVM (Kilo Virtual Machine)
- Configurations and Profiles
  - CLDC (Connected Limited Device Configuration)
  - MIDP (Mobile Information Device Profile)
- Examples
- Availability and Resources

http://www.cis.uoguelph.ca/~qmahmoud
Web Content for Mobile Devices

- Markup languages to deliver Web content to device browsers:
  - HDML
    - Phone.com (now Openwave)
  - Compact HTML (cHTML)
    - NTT DoCoMo’s i-mode network
  - WAP Forum’s WML
    - An emerging standard for content delivered to mobile devices
WAP Network Structure

The WAP Gateway plays an important role
WAP Programming Model

Similar to the Web programming model with extensions for the wireless environment
WML Example

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="MyFirstCard" title="First Card">
    <p align="center">
      My First WML Example
    </p>
  </card>
</wml>
Transcoding Proxies

- Transcoding proxies are becoming more capable and widely used.

- HTML, cHTML, and WML are converging towards XHTML.

- XHTML is the re-writing of HTML as an XML-based markup language.
Java 2 Platform

Virtual Machines and horizontal and vertical APIs specified in *configurations* and *Profiles*

Source: j2me white paper
Configurations

A configuration defines the minimum APIs and VM capabilities for a family of devices:

- Similar requirements of memory size and processing capabilities

The minimum APIs that an application developer can expect to be available on implementing devices
Configurations

- May not contain any optional features
- Defined through the Java Community Process (JCP)
  - http://java.sun.com/jcp (www.jcp.org)
- Subject to compatibility tests
Profiles

- A profile is a collection of APIs that supplement a configuration to provide capabilities for a specific vertical market.

- Defined through Java Community Process initiative (www.jcp.org).

- Subject to compatibility tests.
How do they fit together?

Profiles are built on top of configurations

- Host Operating System
- Configurations (CLDC)
- Profiles (MIDP)
- Optional Packages
CLDC

- Targeted at devices with:
  - 160 to 512 KB of total memory available for Java technology
  - Limited power (e.g. battery)
  - Limited connectivity to a network (wireless)
  - Constrained User Interface (small screen)

- It is available for free download

- Reference implementation built using KVM
MIDP

- Targets mobile two-way communication devices implementing the CLDC
- It addresses:
  - Display toolkit (user input)
  - Persistent data storage
  - HTTP based networking using CLDC generic connection framework
- Available for free download
KVM

- Stands for Kilo Virtual Machine
- Originated from a research project called **Spotless** at Sun Research Labs
- Implements the classes defined in the CLDC specification + some additional UI classes
- **Note:** the UI classes are not part of the CLDC and can be removed at any time
KVM...

- A complete runtime environment for small devices
- Built from the ground up in C
- Small footprint (40 – 80 KB)
- Class file verification takes place off-device
- Supports multi-threading
- Supports garbage collection
KVM Security

- VM level security
  - Off-device pre-verification
  - Small in-device verification

- Application level security
  - No Security Manager
  - Sandbox security model:
    - Applications run in a closed environment
    - Applications can call classes supported by the device
KVM

- It runs on Solaris, Win32, and PalmOS
  - MIDP4Palm (java.sun.com/products/midp4palm)
Wireless Device Stack

- MIDlet
- Other Profiles
- MIDP
- CLDC APIs
- KVM
- Host OS
- Train Schedule
- UI
- HTTP
- No floats
CLDC Internals

The CLDC specification specifies VM features required by a CLDC implementation

- Specifies requirements and APIs for
  - Input/Output
  - Networking
Beyond the CLDC scope

- Profiles implemented on top of CLDC specify APIs for:
  - User Interface support
  - Event handling
  - Persistent support
  - High-level application model
- An example profile is the Mobile Information Device Profile (MIDP)
Language & VM Compatibility

Goal:

- Full java language and VM specification compatibility

Language-level exception:

- No floating point support in CLDC 1.0
  - No hardware floating point support
  - Manufacturers and developers can include their own floating point
CLDC vs. J2SE JVM

Limitations in CLDC supporting JVM:

- No floating point support
- No finalization
- Limited error handling
- No Java Native Interface (JNI)
- No support for reflection
- No thread groups or daemon threads
- No weak references
CLDC APIs

- Classes inherited from J2SE v1.3 are in packages:
  - java.lang
  - java.io
  - java.util

- New classes introduced by the CLDC are in package:
  - javax.microedition
CLDC Libraries: java.lang.*

- Boolean
- Byte
- Character
- Class
- Integer
- Long
- Math
- Object
- Runnable
- Runtime
- Short
- String
- StringBuffer
- System
- Thread
- Throwable
CLDC Libraries: java.io.*

- ByteArrayInputStream
- ByteArrayOutputStream
- DataInput
- DataOutput
- DataInputStream
- DataOutputStream
- InputStream
- OutputStream
- InputStreamReader
- OutputStreamWriter
- PrintStream
- Reader
- Writer
CLDC Libraries: java.util.*

- Calendar
- Date
- Enumeration
- Hashtable
- Random
- Stack
- TimeZone
- Vector
MIDP internals

Goal:
- MIDP implementation must fit in small footprint (128KB ROM)
- Must run with limited heap size (32-200KB RAM)

To be implemented by device manufacturers, operators, or developers
MIDlets

A MIDlet consists of a class that extends the MIDlet class and other classes as needed.

To handle events it must implement the CommandListener interface.

```java
public class MyMIDlet extends MIDlet implements CommandListener {
}
```
MIDP Application Lifecycle

- MIDlets move from state to state in the lifecycle:
  - **Start**: acquire resources and start executing
  - **Pause**: release resources and wait
  - **Destroyed**: release all resources and end all activities
MIDlet Packaging

- Two or more MIDlets form a MIDlet suite
- One or more MIDlets may be packaged in a single JAR file that includes:
  - A manifest describing the contents
  - Java classes for the MIDlet(s)
  - Resource file(s) used by the MIDlet(s)
- Each jar file is accompanied by a Java Application Descriptor (JAD) file
MIDlet Packaging

Java Application Descriptor (JAD) file provides info:

- Configuration properties
- Pre-download properties
  - Size, version, storage requirements
import javax.microedition.midlet.MIDlet;
import javax.microedition.lcdui.*;

public class FirstMIDlet extends MIDlet {
    Display display = null;
    TextBox tb = null;
    public FirstMIDlet() {
        display = Display.getDisplay(this);
    }
}
Example MIDlet ...

```java
public void startApp() {
    tb = new TextBox("FirstMIDlet", "Welcome to MIDP Programming", 40, 0);
    display.setCurrent(tb);
}

public void pauseApp() {}

public void destroyApp(boolean unconditional) {
}
```
Example MIDlet...

- Compile (javac)
- Preverify (off device preverification)
- Create a JAR file: first.jar
- Create a JAD file: first.jad
  - MIDlet-Name: MyFirst
  - MIDlet-Version: 1.0.0
  - MIDlet-Vendor: Sun Microsystems, Inc.
  - MIDlet-Description: My First MIDlet
  - MIDlet-Info-URL: http://java.sun.com/j2me/
  - MIDlet-Jar-URL: first.jar
  - MIDlet-Jar-Size: 1063
  - MicroEdition-Profile: MIDP-1.0
  - MicroEdition-Configuration: CLDC-1.0
  - MIDlet-1: MyFirst, FirstMIDlet
Example MIDlet: Testing

midp -descriptor first.jad

http://www.cis.uoguelph.ca/~qmahmoud
**MIDlet Example: Deploy**

- To deploy a MIDlet on a web server, you need to add a new MIME type:

  `text/vnd.sun.j2me.app-descriptor jad`

- Use the following command to run:

  `midp -transient http://hostname/path/first.jad`
Simplifying the Development Effort

Sun’s Wireless Toolkit

http://www.cis.uoguelph.ca/~qmahmoud  Qusay H. Mahmoud 39
MIDP APIs

The MIDP specifies APIs for:

- User Interface
- Networking (based on CLDC)
- Persistent Storage
- Timers
MIDP User Interface

- Not a subset of AWT or Swing because:
  - AWT is designed for desktop computers
  - Assumes certain user interaction models (pointing device such as a mouse)
  - Window management (resizing overlapping windows). This is impractical for cell phones

- Consists of high-level and low-level APIs
MIDP UI APIs

- High-level API
  - Applications should be runnable and usable in all MIDP devices
  - No direct access to native device features

- Low-level API
  - Provide access to native drawing primitives, device key events, native input devices
  - Allows developers to choose to compromise portability for user experience
MIDP UI Programming Model

- The central abstraction is a **screen**
- Only one screen may be visible at a time
- Three types of screens:
  - Predefined screens with complex UI components (List, TextBox)
  - Generic screens (Form where you can add text, images, etc)
  - Screens used with low-level API (Canvas)
MIDP UI and Display

- The Display class is the display manager.
- It is instantiated for each active MIDlet.
- Provides methods to retrieve information about the device’s display capabilities.
- A screen is made visible by calling: `Display's setCurrent(screen);`
MIDP UI Classes

- **javax.microedition.lcdui classes:**
  - Alert, AlertType, Canvas, ChoiceGroup,
  - Command, DateField, Display, Displayable, Font,
  - Form, Gauge, Graphics, Image, ImageItem,
  - Item, List, Screen, StringItem, TextBox,
  - TextField, Ticker

- **javax.microedition.lcdui interfaces:**
  - Choice, CommandListener, ItemStateListener
MIDP UI Class Diagram

Major classes and interfaces:

- **Object**
- **Displayable**
  - 0..n **Command**
- **Screen**
- **Canvas**
- **Alert**
- **List**
- **Form**
- **TextBox**
- **ChoiceGroup**
  - **DateField**
  - **Gauge**
  - **ImageItem**
  - **StringItem**
  - **TextField**
High-Level API Examples

- **List:**
  ```java
  Display display = Display.getDisplay(this);
  List menu = new List("Method of payment", Choice.EXCLUSIVE);
  menu.append("Visa");
  menu.append("MasterCard");
  menu.append("Amex");
  display.setCurrent(menu);
  ```
High-Level API Examples...

**Form (Date/Time info):**

```java
DateField date = new DateField("Today’s date", DateField.TIME);
Form form = new Form("Date Info");
form.append(date);
display.setCurrent(form);
```
High-Level Examples...

- Form (Sign in screen):
  Display display = Display.getDisplay(this);
  TextField userName = new TextField("LoginID:", "", 10, TextField.ANY);
  TextField password = new TextField("Password:", "", 10, TextField.PASSWORD);
  Form form = new Form("Sign in");
  form.append(userName);
  form.append(password);
  display.setCurrent(form);
Low-level Example

Canvas:

```java
public class MyCanvas extends Canvas {
    public void paint(Graphics g) {
        g.setColor(255, 0, 0);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(255, 255, 255);
        g.drawString("Hello World!", 0, 0, g.TOP | g.LEFT);
    }
}
```
Low-level Example...

- Instantiate and display MyCanvas

```java
public class MyMidlet extends MIDlet {
    public MyMidlet() { // constructor
    }
    public void startApp() {
        Canvas canvas = new MyCanvas();
        Display display = Display.getDisplay(this);
        display.setCurrent(canvas);
    }
    // pauseApp() and destroyApp()
}
```

http://www.cis.uoguelph.ca/~qmahmoud
Qusay H. Mahmoud
Input Handling

- High-Level API input is handled using abstract commands
  - No direct access to soft buttons
  - Commands are mapped to appropriate soft buttons or menu items
Input Handling: Example

**TextBox screen with commands:**

```java
Display display = Display.getDisplay(this);
TextBox tb = new TextBox("MIDP", "Welcome to MIDP Programming", 40, TextField.ANY);
Command exit = new Command("Exit", Command.SCREEN, 1);
Command info = new Command("Info", Command.SCREEN, 2);
Command buy = new Command("Buy", Command.SCREEN, 2);
tb.addCommand(exit);
tb.addComment(info);
tb.addCommand(buy);
display.setCurrent(tb);
```

http://www.cis.uoguelph.ca/~qmahmoud

Qusay H. Mahmoud
Event Handling: High-level

**High-level Events:**

- Based on a listener model
- Screen objects can have listeners for commands
- For an object to be a listener, it must implement the `CommandListener` interface
- This interface has one method: `commandAction`
Event Handling: Example

- MIDlet implements CommandListener

```java
public class MyMIDlet extends MIDlet implements CommandListener {
    Command exitCommand = new Command(...);
    // other stmts

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

Qusay H. Mahmoud
Event Handling: Example

Handling List events:

```java
public void commandAction(Command c, Displayable d) {
    if (c == exitCommand) { ..

} else {
    List down = (List)display.getCurrent();
    switch(down.getSelectedIndex()) {
        case 0: testTextBox();break;
        case 1: testList();break;
        case 2: testAlert();break;
        case 3: testDate();break;
        case 4: testForm();break;
    }
}
Event Handling: Low-level

- Low-level Events:
  - Low-level API gives developers access to key press events
  - Key events are reported with respect to key codes
  - MIDP defines key codes: KEY_NUM0 .. KEY_NUM9, KEY_STAR, KEY_POUND
Handling Events: example

- Low-level events

protected void keyPressed(int keyCode) {
    if (keyCode > 0) {
        System.out.println("keyPressed 
        +getGameAction(keyCode));
    } else {
        System.out.println("keyPressed 
    }
}
MIDP UI Design Principles

- Make the UI simple and easy to use
- Use the high-level API (portability)
- If you need to use low-level API, keep to the platform-independent part
- MIDlets should not depend on any specific screen size
- Entering data is tedious, so provide a list of choices to select from
Networking

- J2SE and J2EE networking APIs are not suitable for handheld devices
  - Require several megabytes of memory to run
  - Device manufacturers who work with circuit-switched networks require TCP support
  - Device manufacturers who work with packet-switched networks require UDP support
  - Other devices have specific mechanisms for communications
CLDC Generic Connections

- A set of related abstractions at the programming level
- No abstractions for different forms of communications
- All connections are created using the `Connector.open()`
- If successful, it returns an object that implements one of the generic connection interfaces
Connection Interfaces

Connection

StreamConnectionNotifier

DatagramConnection

InputConnection

OutputConnection

StreamConnection

ContentConnection
Example Connections

- **HTTP:**
  
  `Connector.open("http://www.host.com");`

- **Socket:**
  
  `Connector.open("socket://host.com:80");`

- **Datagram:**
  
  `Connector.open("datagram://address:port");`

- **File:**
  
  `Connector.open("file://myfile.txt");`
Advantages of CLDC

Generic Connections

- Isolate the differences between the setup of one protocol and another
- Most of the application code remains the same regardless of the protocol you use

**Note:** CLDC itself does not provide any protocol implementation
MIDP Connectivity

- It provides support for HTTP (HttpURLConnection)

- Why? HTTP can be implemented using IP protocols or non-IP protocols
HttpConnection

- Part of the javax.microedition.io
- Defines the necessary methods and constants for an HTTP connection

```java
HttpConnection c = (HttpConnection) Connector.open("http://quotes.yahoo.com");
C.setRequestMethod(HttpConnection.POST);
C.setRequestProperty("Content-Language", "en-CA");
```
Invoking Remote Applications

A MIDlet may invoke remote applications:

- Fetching a page
- Invoking a CGI script (GET or POST method)
- Invoking a Servlet
Example: Invoke a CGI Script

- GET Method:

```java
String url = "http://host/cgi-bin/getgrade?idnum=182061";
c = (HttpConnection) Connector.open(url);
c.setRequestMethod(HttpConnection.GET);
// set some request properties: c.setRequestPropert(" ", " ");
is = c.openDataInput();
while((ch = is.read()) != -1) {
    b.append((char)ch);
}
```

http://www.cis.uoguelph.ca/~qmahmoud

Qusay H. Mahmoud
Example...

If you want to send data to a remote application:

```java
String s = "stuffToSend";
byte postmsg[] = s.getBytes();
for(int i=0; i<postmsg.length; i++) {
    os.writeBytes(postmsg[i]);
}
// OR
os.write(s.getBytes());
```
Servlet as a Mediator

- Devices with no IP stack may access and use ‘net services through a servlet

![Diagram](http://www.cis.uoguelph.ca/~qmahmoud)
Databases

- A persistent storage: a place to store the state of objects
- Facilities provided in J2SE and J2EE are not suitable for handheld devices
- MIDP provides a record-oriented database mechanism to persistently store data and retrieve it later
MIDP’s RMS

- Lightweight record-oriented database
  - Device independent API
  - Unique recordID for each record within the store
  - A record is an array of bytes
  - Shared within MIDlet suite
  - Support for enumeration, sorting, and filtering

javax.microedition.rms
MIDP RMS Methods

- **Record Store**
  - openRecordStore, closeRecordStore,
  - listRecordStore, deleteRecordStore,
  - getRecordSize, getNumRecords

- **Record Data**
  - addRecord, deleteRecord, getRecord, setRecord,
  - getRecordSize

- **Record Selection**
  - RecordEnumeration, RecordFilter,
  - RecordComparator
RMS: Record Stores

- To open a record store:
  ```java
  RecordStore db = RecordStore.openRecordStore("myDB", true);
  ```

- To close a record store:
  ```java
  db.closeRecordStore();
  ```
Create/Add a new record

To create a new record:

ByteArrayOutputStream baos = new ByteArrayOutputStream()
DataOutputStream dos = new DataOutputStream(baos);
dos.writeUTF(record);
Byte b[] = baos.toByteArray();
db.addRecord(b, 0, b.length);
Building a Stock Database

See appendix for source code:

- Stock.java
- StockDB.java
- QuotesMIDlet.java
StockQuotes MIDlet

Choose One:
Stock Quotes

Choose Stocks
🌞 SUNW @ 15.19
🌞 MOT @ 14.25
☐ AOL @ 50.90
☐ IBM @ 113.60

Enter a Stock Name:
SUNW

Enter a Stock Name:
MOT

Enter a Stock Name:
AOL

Enter a Stock Name:
IBM

Database Application
List Stocks
Add A New Stock

About
Exit

http://www.cis.uoguelph.ca/~qmahmoud
Qusay H. Mahmoud
MIDP Timers

- Handle queuing and delivery

- Timer task:
  - Multiple tasks per timer
  - Periodic
  - Fixed interval
  - One-time execution
Tasks

To define a task, create a subclass of TimerTask:

```java
import java.util.*;
public class MyTask extends TimerTask {
    public void run() {
        System.out.println(“Run Task”);
    }
}
```
Tasks...

Schedule it for execution by creating a Timer object and invoking schedule() in Java:

```java
Timer timer = new Timer();
TimerTask task = new MyTask();
// wait five seconds before executing
timer.schedule(task, 5000);
// wait two seconds before executing then
// execute every five seconds
timer.schedule(task, 2000, 5000);
```
MIDP Next Generation

MIDP NG will:

- Maintain backward compatibility with MIDP 1.0
- Continue focus on wireless phones
- Maintain small footprint (limit API growth)
- Fine tune MIDP 1.0 APIs
- Enable mobile commerce
MIDP Next Generation

Areas to investigate:

- HTTPS and secure networking (SSL)
- Network connectivity via sockets and datagrams
- Formal inclusion of Over The Air Provisioning
- Inclusion of a small XML Parser
- Sound API
MIDP for Palm OS >= 3.5

- A J2ME application runtime environment based on CLDC 1.0 and MIDP 1.0.
- It is targeted at handheld devices (such as Palm Pilot, Handspring Visor) running Palm OS 3.5 or higher.
- Java Manager: MIDP.PRC
MIDP for Palm OS

To install:
- Place palm device in the cradle
- Use Palm Desktop Software to install MIDP.PRC
MIDP for Palm OS

- Comes with a PRC converter tool
  - C:\midp4palm\java -jar Converter.jar

http://www.cis.uoguelph.ca/~qmahmoud  Qusay H. Mahmoud  85
MIDP for Palm OS

- Install GUI.PRC
MIDP for Palm OS

To have more control, use command line tool:

- Converting a single MIDlet JAR to PRC file
  ```
  java -jar MakeMIDPApp.jar -nobeam -o Stocks.prc -JARtoPRC StockQuotes.jar
  StockMIDlet
  ```

- Converting a MIDlet suite to PRC file
  ```
  java -jar MakeMIDPApp.jar -jad MySuite.jad -o MySuite.prc -JARtoPRC MySuite.jar MySuite
  ```
MIDP for Palm OS

Networking and Databases
Other Configurations/Profiles

- PDA Profile (CLDC-based)
- Connected Device Configuration (CDC)
  - CVM
  - Supports full Java library
- Foundation Profile
- Personal Profile
- RMI Profile
PDA Profile

- www.jcp.org/jsr/detail/75.jsp
- Based on the CLDC 1.0
- Will provide user interface and data storage APIs for handheld devices
- The UI API is expected to be a subset of the AWT
- No reference implementation yet
CDC/Foundation

- CDC targets the next generation of wireless, handheld consumer devices
- The Foundation Profile is meant to serve as a foundation for other profiles
- It extends the CDC by adding most of the missing J2SE core libraries, except those related to UI API's for beans, rmi, sql are not part of CDC/Foundation
- Reference Implementations are available for Linux and VxWorks
CLDC vs. CDC

- CLDC implements a subset of Java features and APIs
- KVM
- For limited devices
- 16 or 32-bit processors
- Targets devices with 160-512 KB of memory

- CDC is a full Java implementation
- CVM
- For more powerful devices
- 32-bit processors
- Targets devices with at least 2 MB of memory

http://www.cis.uoguelph.ca/~qmahmoud
Personal Java is being redefined as the Personal Profile
- It extends the Foundation Profile
- Provides GUI capable of running Java web applets
- Backward compatible with 1.1 and 1.2 PersonalJava applications
- No reference implementation yet
RMI Profile

- Extends the CDC and Foundation to provide Remote Method Invocation for devices
- Therefore, it is meant to be used with the CDC/Foundation and not CLDC/MIDP
- Requires TCP/IP network connectivity
- Compatible with J2SE RMI API 1.2.x or higher
- No reference implementation yet
Availability and Resources

- Sun’s CLDC implementation supports development using Solaris, Win32, PalmOS
- NTT DoCoMo started Java-based cell phone service in January/01
- Motorola announced a Java-enabled GSM phone (Accompli A008).
- Motorola & Nextel (i50sx, i85s)
- Nokia, Ericsson, and others support J2ME
J2ME SDKs

- Sun’s Wireless Toolkit: java.sun.com/products/sjwtoolkit
- NetBeans (Mobility Pack): http://www.netbeans.org
- RIM’s BlackBerry JDE: www.rblackberry.com/developers
J2ME Resources

- J2ME (Java ME):
  http://java.sun.com/j2me

- CLDC and KVM:
  http://java.sun.com/products/cldc

- MIDP:
  http://java.sun.com/products/midp

- Wireless Toolkit:
  http://java.sun.com/products/sjwtoolkit
Additional Resources

- Sun Microsystems Developer Network:
  java.sun.com/wireless
- WirelessDevNet Developer Portal:
  www.wirelessdevnet.com
Thank you

Q&A
Copyright Info

©Copyright 2001 Qusay H. Mahmoud  
All rights reserved.

Java™ and J2ME™ are registered trademarks of Sun Microsystems, Inc.  
All trademarked product and company names are the property of their respective trademark holders.