J2ME Programming - Part I

Jeffrey Peacock

jeffp@JeffreyPeacock.com

Introduction

- What is J2ME?
- What are all the acronyms?

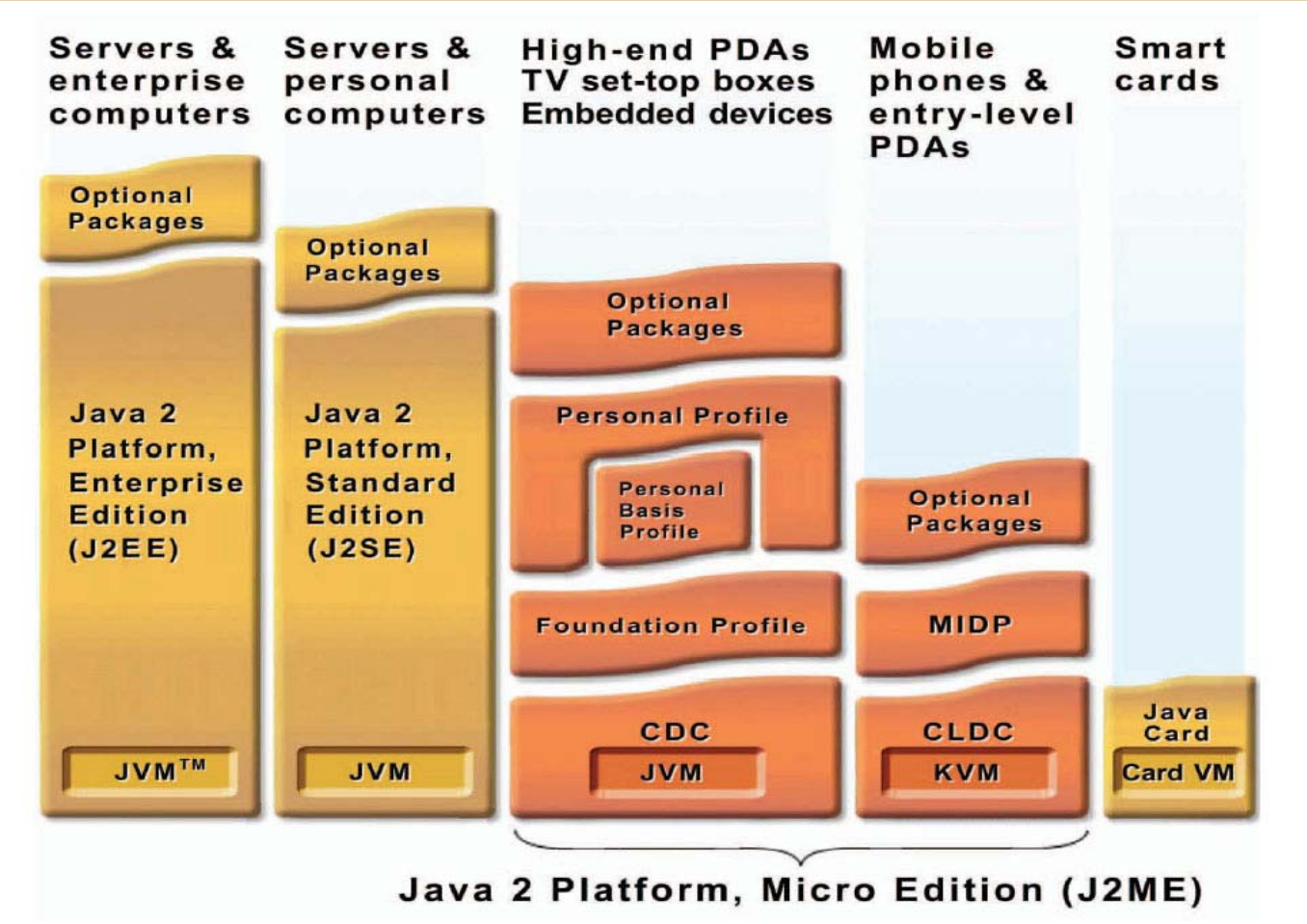
<u>Observation:</u>

Crossing over to J2ME from the cushy world of J2SE is a pain in the ass

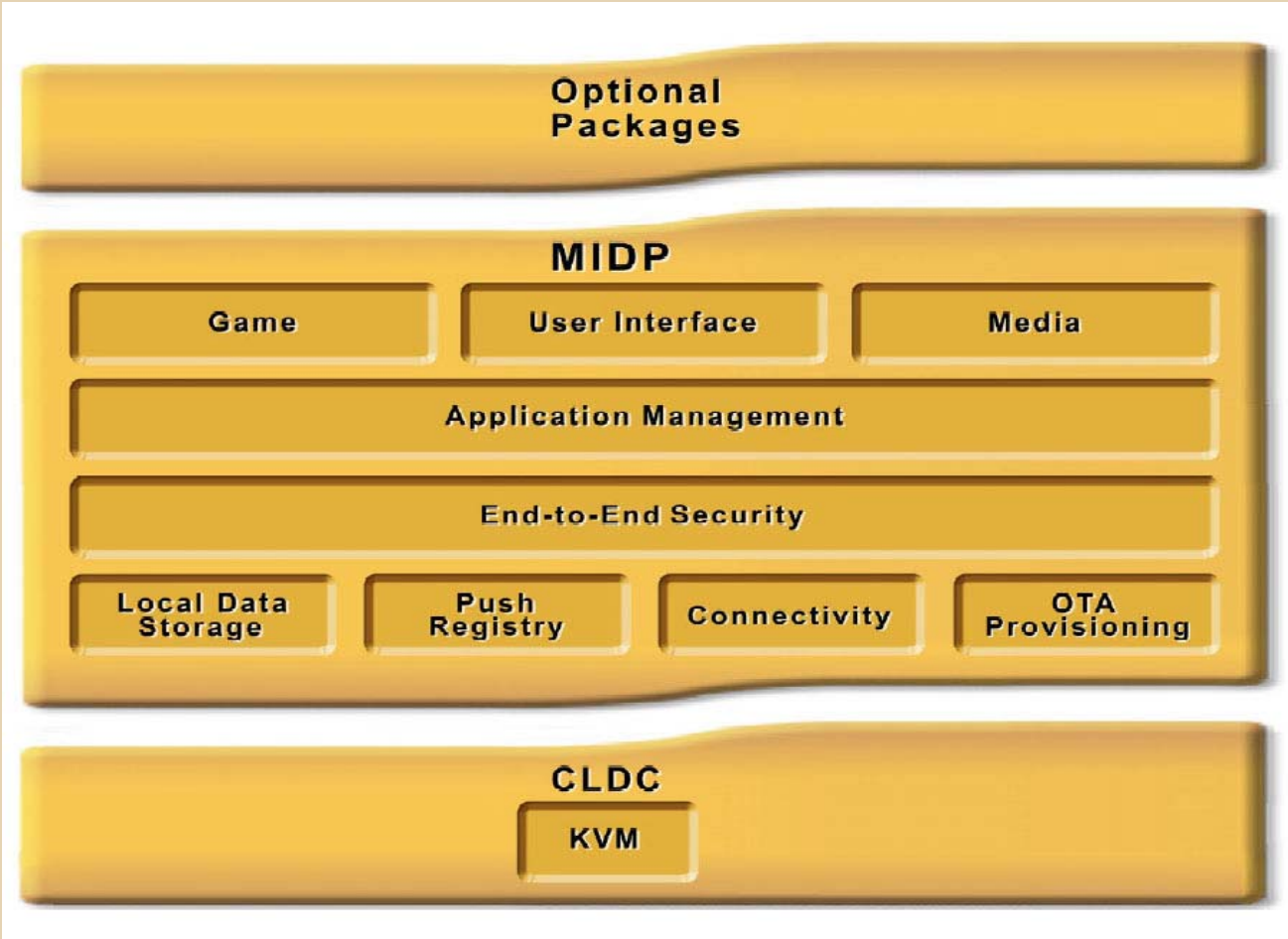
Agenda

- J2ME's Alphabet Soup
  - CLDC, CDC, MIDP, etc.
- Configurations & Profiles
- CLDC Details
- MIDP Details
- Examples

Dangerous & Confusing Java World



CLDC/MIDP



CLDC

- Connected Limited Device Configuration
  - Defines portable, minimum-footprint Java for small resource-constrained, connected devices, e.g.:
    - 160-512kb total memory available for Java
    - 16-bit or 32-bit processor
    - Low power consumption – likely battery powered
    - Network connectivity – wireless, intermittent, limited bandwidth
- Essentially the VM for a small device

CLDC (cont.)

- Target devices:
  - Cell Phones
  - Two-way pagers
  - PDA's (low-end)
  - Organizers,
  - Home appliances
  - Point-of-sale terminals
  - Others …

CLDC Java Compatibility 1.0

- A VM Conforming to the Goals of the CLDC is Only Possible by Limiting the Language:

• No floating point support • No finalization • No JNI • No user defined class loaders	• No reflection • No thread groups or daemon threads • No weak references • Limited error handling

CLDC Java Compatibility 1.0 (cont.)

Limitations (cont.):

- Security Model
  - Class File Verification is done off-device using a *preverifier* which annotates the class files.
  - Runtime verification is done via stack maps utilizing the annotations
  - Classes deployed for download must be preverified

CLDC 1.0: J2SE Derived Classes

- System Classes

• **java.lang.Object** • **java.lang.Class** • **java.lang.Runtime** • **java.lang.System** • **java.lang.Thread** • ***java.lang.Runnable*** • **java.lang.String**	• **java.lang.Stringbuffer** • **java.lang.Throwable**

CLDC 1.0: J2SE Derived Classes (cont.)

- Data type and collection classes

• **java.lang.Boolean** • **java.lang.Byte** • **java.lang.Short** • **java.lang.Integer** • **java.lang.Long** • **java.lang.Character** • **java.util.Vector** • **java.util.Stack**	• **java.util.Hashtable** • ***java.lang.Enumeration***

CLDC 1.0: J2SE Derived Classes (cont.)

- Input/output classes

• java.io.InputStream • java.io.OutputStream • java.io.ByteArrayInputStream • java.io.ByteArrayOutputStream • java.io.InputStreamReader • java.io.OutputStreamWriter • java.io.DataInputStream • java.io.DataOutputStream	• java.io.DataInput • java.io.DataOutput • java.io.Reader • java.io.Writer • java.io.PrintStream

CLDC 1.0: J2SE Derived Classes (cont.)

- Calendar & time; additional

• **java.util.Calendar** • **java.util.Date** • **java.util.TimeZone**	• **java.util.Random** • **java.lang.Math**

CLDC 1.0: J2SE Derived Classes (cont.)

- Furthermore:
  - Classes are derived from J2SE 1.3
  - Less classes, Less exceptions
  - Limited `Error`'s
  - Limited I18N Unicode support
  - Limited VM properties but no `Properties` class so no discovery by `Enumeration`

CLDC 1.0: Specific Classes

- Generic Connection Framework

javax.microedition.io.

- **Connection**
- ***ContentConnection***
- ***Datagram***
- ***DatagramConnection***
- ***InputConnection***
- ***OutputConnection***
- ***StreamConnection***
- ***StreamConnectionNotifier***

CLDC 1.0: Generic Connection Framework

- Addresses six basic interface types
  1. Basic serial input device
  2. Basic serial output device
  3. Datagram oriented device
  4. Circuit oriented device
  5. Client-server connection notification mechanism
  6. Basic web server connection

CLDC 1.0: Generic Connection Framework (cont.)

- General Form

```
Connector.open (“<protocol>:<addr>;<parms>”);
```

  - HTTP

```
Connector.open (“http://www.sun.com”);
```

  - Sockets

```
Connector.open
  (“socket://192.144.111.222:2800”);
```

  - Serial Port

```
Connector.open (“comm:0;baudrate=9600”);
```

  - Datagrams, Files, …

CLDC 1.0: Generic Connection Framework (cont.)

- Does not define any protocol implementations
  - This is the responsibility of the Profile

CLDC 1.1

- Floating point support; classes Float & Double
- Subset of J2SE weak reference support
- Calendar, Date and TimeZone redesigned to be more J2SE compliant
- Error handling requirements clarified
- Thread object are more like J2SE
- Min. memory budget raised from 160 -> 190kb

CLDC 1.1 (cont.)

- Various new methods on existing classes
- Specification clarifications
- More detailed verifier specification

Problem:

Today's available devices are typically CLDC 1.0

MIDP

- Mobile Information Device Profile
  - Defines device-type-specific sets of API's for a particular vertical market or industry

MIDP (cont.)

- Minimum Hardware Requirements

Memory • 128kb NV RAM • 8kb NV RAM for persistent data • 32kb RAM for VM runtime Display • Screen-size: 96x54 • Display depth: 1-bit • Pixel shape: 1:1	Input (one or more) • “one-handed-keypad” • “two-handed-keypad” • touch screen Networking • Two-way, wireless, possibly intermittent, limited bandwidth *** Any device with these capabilities can host the MIDP.*

MIDP (cont.)

- Added Functionality
  - User interface support (LCDUI)
  - Networking support (HTTP based)
  - Persistent storage (RMS, record based, not file)
  - Misc. classes such as timers and exceptions

MIDP 1.0: Specific Classes

- Additions to CLDC

java.util
- Timer
- TimerTask

java.microedition.io
- *HttpConnection*

MIDP 1.0: Specific Classes

- MIDP Unique

javax.microedition.midlet

- **MIDlet**

javax.microedition.rms

- **RecordStore**
- ***RecordComparator***
- ***RecordEnumeration***
- ***RecordFilter***
- ***RecordListener***

MIDP 1.0: Specific Classes

MIDP 1.0: Specific Classes (cont.)

javax.microedition.lcdui

- ***CommandListener***
- ***ItemStateListenerAlert***
- **AlertType**
- **Canvas**
- **ChoiceGroup**
- **Command**
- **DateField**
- **Display**
- **Displayable**
- **Font**
- **Form**
- **Gauge**
- **Graphics**
- **Image**
- **ImageItem**
- **Item**
- **List**
- **Screen**
- **StringItem**
- **TextBox**
- **TextField**
- **Ticker**

MIDP User Interface: LCDUI

- Limited Connection Device User Interface
  - This is not your daddy's SWING (or AWT)
  - Two types of **`Displayable`** objects:
    - **`Canvas`**: low-level objects, provide graphics and handle input
    - **`Screen`**: high-level objects, user interface components – e.g., **`Alert`**, **`List`**, **`TextBox`**, **`Form`**
  - Supports PNG (only?)

MIDP Persistence: RMS

- Record Management System
  - Provides *records* and *record stores*
  - records are:
    - An opaque array of bytes
    - Variable length
    - Identified by **`recordId`**
    - Accessible sequentially or directly
  - record stores
    - Can manage allocation/de-allocation as they see fit

MIDlets

- Basic unit of execution and distribution
- Lifecycle of a MIDlet has 3 distinct states:
  1. Paused
  2. Active
  3. Destroyed

Hello World!

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

public class HelloWorld extends MIDlet
 implements CommandListener {
   public HelloWorld () {
    TextBox tb = new TextBox (“Hello World.”, ..);
    …
   }
   protected void startApp() {…}
}
```

MIDlet Suites

- One or more MIDlets in a single JAR file
  - Share common name space for persistent storage
  - Runtime heap
  - `static` fields in classes
- Is the basic unit of application install, update, removal
  - Individual MIDlets, classes, or files cannot be installed, updated or removed

MIDlet Suite Packaging

- MIDlet suite JAR contains:
  - Class files (transitive closure excluding CLDC/MIDP) for all contained MIDlets
  - Resource files (e.g., images, static text, etc.)
  - Manifest
    - Must contain specific properties which become part of the execution environment

MIDlet Suite Packaging (cont.)

- Manifest example:

```
MIDlet-Name: CardGames
MIDlet-Version: 1.1.9
MIDlet-Vendor: Motorola
MIDlet-1: Solitaire, /Solitaire.png, com.motorola.Solitaire
MIDlet-1: Blackjack, /Blackjack.png, com.motorola. Blackjack
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
```

MIDlet Suite Packaging (cont.)

- Application Descriptor
  - Dreaded “JAD” file
  - Separate and optional to the JAR file
  - Allows the device to determine if the associated application is suited to the device before loading the full JAR file
  - Optional but MIDP-compliant implementations must accept

MIDlet Suite Packaging (cont.)

- JAD example

(required)

```
MIDlet-Name: CardGames
MIDlet-Version: 1.1.9
MIDlet-Vendor: Motorola
MIDlet-Jar-URL: http://www.Motorola.com/…
MIDlet-Jar-Size: 10389
```

(optional)

```
MicroEdition-Profile: MIDP-1.0
MicroEdition-Configuration: CLDC-1.0
```

MIDlet Suite Packaging (cont.)

- Manifest and JAD provide rudimentary version control:
  - **`MIDlet-Name`**, **`MIDlet-Version`**, & **`MIDlet-Vendor`** must match
- Distribution is typically done by using the URL of the JAD file

MIDP 2.0

- New features (required)
  - Sound — Mobile Multimedia API (JSR-000135)
  - Security for MIDlet suites — a notion of trust for MIDlet suites, and security domains to protect security-sensitive APIs from untrusted MIDlet suites
  - Games API — a series of classes that enable the development of rich gaming content for wireless devices
  - Additional LCDUI components — new components include a spacer (non-interactive, used to help position other items), a custom item (customizable item that can introduce new visual and interactive elements into a form), and so on.

MIDP 2.0 (cont.)

- New features (required) (cont.)
  - Additional LCDUI features — new features include the ability to add abstract commands to alerts and items, a full-screen mode for the `Canvas` class, vibrate, and so on.
  - Push functionality — ability to launch a MIDlet in response to an incoming message.

MIDP 2.0 (cont.)

- New features (required) (cont.)
  - Additional network protocols — networking support, still based on the Generic Connection framework from the CLDC, contains more predefined protocols:

• Comm	• Socket
• Datagram	• Server Socket
• HTTP	• SSL
• HTTPS	

MIDP 2.0 (cont.)

- Must support MIDP 1.0 MIDlets & suites

Problem:

Like CLDC 1.0, MIDP 1.0 devices are the current norm.

Summary

- Developing for J2ME:
  - Devices have limited features
  - Language and packages have limited features
  - Requires modifying your mental model of “standard” Java development
  - Requires special considerations for integrating, building and deploying with “standard” Java applications

References

- Sun's J2ME home page
  http://java.sun.com/j2me/index.jsp
- Programming Wireless Devices with the Java 2 Platform, Micro Edition, Riggs, et al.