Mobile Game Architecture and Design
J2ME Platform and Tools
In this session...
- J2ME platform
- J2ME architecture
- J2ME development tools
- OTA Provisioning
J2ME Platform – overview

★ Java 2 Standard Edition
  - Standard Client/Server applications including web based applications.

★ Java 2 Enterprise Edition
  - Multi-tiered and potentially distributed applications.
  - Collection of vendor independent APIs for server-side programming.

★ Java 2 Micro Edition
  - Client/Server applications for mobile devices with limited power, network connectivity and GUI capabilities.
  - Goals:
    - Focuses on the personal mobile devices with limited resources and differences in capabilities, features and processing abilities.
    - To Provide facility to connect devices to various types of networks.
    - To provide facility to deliver applications and data over a network connection.
J2ME Platform Organization

- Applications
- Profile
  (Device-family specific classes)
- Optional Packages
- Vendor-specific Classes (OEM)
- Configuration
- Native Operating System
- Device/Hardware

SOURCE: WWW.SUN.COM
### J2ME Platform – Conceptual Layers

<table>
<thead>
<tr>
<th>Example Devices</th>
<th>Virtual Machine Layer</th>
<th>Configuration Layer</th>
<th>Profile Layer</th>
<th>Foundation, Personal, Per. Basis</th>
<th>MIDP, I MP</th>
<th>GSM, Open Platform</th>
<th>CDC</th>
<th>CLDC</th>
<th>Java Card API, Security, RMI</th>
<th>Java Card VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smartcard</td>
<td></td>
<td></td>
<td></td>
<td>Higher end PDAs, Set-top Boxes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Phones</td>
<td></td>
<td></td>
<td></td>
<td>Mobile Phones, Pagers, Industry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pagers, Industry</td>
<td></td>
<td></td>
<td></td>
<td>devices</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mobile Phones</td>
<td></td>
<td></td>
<td></td>
<td>Mobile Phones, Pagers, Industry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pagers, Industry</td>
<td></td>
<td></td>
<td></td>
<td>Mobile Phones, Pagers, Industry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© SoC, NUS
J2ME Configurations

J2SE

CDC

CLDC

J2ME
CDC (Connected Device Configuration)

- Shared, Fixed, Connected Information Devices
- Robust UI Functions
- 2-16 MB Memory Range (RAM and ROM)
- Greater than 32-bit CPU
- Persistent, High Bandwidth Network Connections
- Examples: TV Set Top Boxes, Internet TV’s, Internet Enabled Screen Phones, High End Communicators, Auto Entertainment/Navigation Systems
CLDC (Connected Limited Device Configuration)

➤ Simple UI
➤ 128KB-1MB Memory Range. The virtual machine and the libraries take 128KB of memory.
➤ 16-bit, 32-bit CPU
➤ Low Bandwidth, Intermittent Networks
➤ Generally don’t use TCP/IP
➤ Examples: Low End Cell Phones, Two-Way Pagers, and Palm OS Handholds
Development Tools - Overview

🌟 J2ME Wireless Toolkit (J2ME WTK 2.5) (http://java.sun.com)
  - J2ME API library, Emulator, Compiler
  - No editor (can use free editors like, JCreate LE, Text Pad 4.7.3)
  - Vendor Specific kits based on J2ME WTK:
    » Sony Ericsson SDK 2.2.4 for the Java(TM) ME Platform (http://developer.sonyericsson.com)
    » The Java SDK for S60 3rd Edition platform (http://forum.nokia.com)

🌟 IDE
  - Commercial: JBuilder (borland.com), JDeveloper (oracle.com)
  - Open Source: eclipse (http://www.eclipse.org), Netbeans 5.0 (www.netbeans.org)
  - Plug-ins for j2me
    » Jbuilder Mobile Set
    » Netbeans Mobility Pack 5.0
Development Tools - Overview

★ 3D Modeling
- Commercial: 3D Studio Max, Maya, Softimage, Litewave 3D
  - Plug-in for J2ME
    » M3g-export.py – (www.nelson-games.de/bl2m3g/)
    - Needs python 2.4.3 (www.python.org)

★ Game Server
- Client Server Communication APIs (such as, J2SE/J2EE based Network API, .Net or WinSock API, SNAP)
- Simple Database System
J2ME WTK & Project Settings

```
Project "demos" loaded
Running with storage root DefaultColorPhone
```

### Required

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDlet-Jar-Size</td>
<td>231598</td>
</tr>
<tr>
<td>MIDlet-Jar-URL</td>
<td>Audiodemo.jar</td>
</tr>
<tr>
<td>MIDlet-Name</td>
<td>AudioSamples</td>
</tr>
<tr>
<td>MIDlet-Vendor</td>
<td>Sun Microsystems, Inc.</td>
</tr>
<tr>
<td>MIDlet-Version</td>
<td>2.0</td>
</tr>
<tr>
<td>MicroEdition-Configuration</td>
<td>CLDC-1.0</td>
</tr>
<tr>
<td>MicroEdition-Profile</td>
<td>MIDP-2.0</td>
</tr>
</tbody>
</table>
**Project Settings**

### User Defined

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBall-MIDI-URL</td>
<td>resource:/audio/pattern.midi</td>
</tr>
<tr>
<td>BBall-wav-URL</td>
<td>resource:/audio/test-wav.wav</td>
</tr>
<tr>
<td>MixTestURL</td>
<td>resource:/audio/test-wav.wav</td>
</tr>
<tr>
<td>PlayerTitle-1</td>
<td>Simple Tone</td>
</tr>
<tr>
<td>PlayerTitle-2</td>
<td>Bark.wav</td>
</tr>
<tr>
<td>PlayerTitle-3</td>
<td>Ring Tone.jar</td>
</tr>
<tr>
<td>PlayerTitle-4</td>
<td>JavaOne Theme.jar</td>
</tr>
<tr>
<td>PlayerTitle-5</td>
<td>JavaOne Theme [http]</td>
</tr>
<tr>
<td>PlayerURL-1</td>
<td>simple tone</td>
</tr>
<tr>
<td>PlayerURL-2</td>
<td>mis:/audio/bark.wav</td>
</tr>
<tr>
<td>PlayerURL-3</td>
<td>resource:/audio/beethoven.js</td>
</tr>
<tr>
<td>PlayerURL-4</td>
<td>resource:/audio/test-wav.wav</td>
</tr>
<tr>
<td>PlayerURL-5</td>
<td><a href="http://java.sun.com/products/javame/media/">http://java.sun.com/products/javame/media/</a>...</td>
</tr>
</tbody>
</table>

### MIDlets

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
<th>Icon</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIDlet-1</td>
<td>Audio Player</td>
<td>/icons/App.png</td>
<td>example.audiodemo.AudioPlayer</td>
</tr>
<tr>
<td>MIDlet-2</td>
<td>Bouncing Ball</td>
<td>/icons/App.png</td>
<td>example.audiodemo.BBall</td>
</tr>
<tr>
<td>MIDlet-3</td>
<td>Mix Demo</td>
<td>/icons/App.png</td>
<td>example.audiodemo.MixDemo</td>
</tr>
</tbody>
</table>
Project Settings

API Selection

Target Platform: JTWI

Profiles
- MIDP 2.0

Configurations
- CLDC 1.0
- CLDC 1.1

Optional
- Wireless Messaging API 1.1 (JSR 120)
- Mobile Media API (JSR 135)

Additional APIs
- Wireless Messaging API 2.0 (JSR 205)
- JAXP XML Parser (JSR 172)
- Bluetooth/CDB (JSR 62)
- Location API for J2ME (JSR 179)
- SATSA-JCMRI (JSR 177)
- SATSA-CRYPTO (JSR 177)
- Mobile Internationalization API (JSR 238)
- SIP API (JSR 180)
- Scalable 2D Vector Graphics API (JSR 226)
- Web Services API (JSR 172)
- PDA Profile for J2ME (JSR 75)
- Mobile 3D Graphics for J2ME (JSR 184)
- SATSA-APDU (JSR 177)
- SATSA-PKI (JSR 177)
- Content Handler API (JSR 211)
- Payment API (JSR 229)
- Advanced Multimedia Supplements (JSR 234)

Folder Structure for HelloWorld project
Pre-verification, Packaging & Deployment

★ The process of doing class verification before deploying the application into the mobile device is referred to as Pre-verification.

★ While you ‘package’ (create JAR & JAD files) the application in a desktop for deployment into a mobile device, the class verification takes place and it creates a pre-verification file and packaged together with the application.

★ Deployment:
  – BlueTooth, IrDA, Data cable….
  – Operator, aggregator/publisher, developer, OTA Provisioning (discussed later…),
Pre-verification, Packaging & Deployment

1. Create Project
   ★ project folder and; bin, src, res and lib sub folders)
   ★ Add source code, resources and additional libraries
     .java,.m3g,.png.,mpg.,wma., mp3, ui library

2. Compile/Build
   * .class files

3. Package (create JAD & JAR for deployment)
   • Normal package
   • Obfuscated package (uses proguard)
     • http://proguard.sourceforge.net
     • Download proguard.zip and extract the JAR file in it into WTK\bin folder
Netbeans IDE features – Designer View

Flow Designer: MIDP-2.0

- Mobile Device
- helloForm [Form]
- logoForm [Form]
Netbeans IDE features – Adding new Form & Navigation Setting

Flow Designer: MIDP-2.0
Netbeans IDE features – User Friendly Form Design

Screen Designer: logoForm [Form]

Device Screen

Assigned Command: backCommand
display hello

Assigned Item
No command

Properties
- Label: imageItem1
- Image: image1
- Alternate Text: null
- Appearance: PLAIN
- Layout: 0x0
- Preferred Size: [Unlocked, Unlock...]
- Default Command: [None]
- Code Properties
  - Instance Name: imageItem1
  - Lazy Initialized: ✔
  - Pre-Init User Code
  - Post-Init User Code

imageItem1 [ImageItem]
javax.microedition.lcdui.ImageItem
MIDP Application Lifecycle (MIDlet Lifecycle)

- MIDlet – is a J2ME-MIDP application. Extends MIDlet class defined in javax.microedition.MIDlet
- MIDlet Suite – collection of MIDlets
- MIDlet suits can share information, are packaged & deployed together as a single JAR file.

MIDP Application Lifecycle →
OTA Provisioning
Simplest form:

Both Client and Server should use the same DA protocol.
DA protocol of MIDP OTA is HTTP

source: developers.sun.com
OTA Provisioning

★ An OTA provisioning system typically encompasses
  - content publication and management,
  - access control,
  - installation (and upgrading) of applications,
  - and tracking the use of applications (content) for billing purposes.
OTA provisioning

Handset with Discovery Application

Wireless Network

OTA Provisioning Portal with Web Server and Presentation System

Back-end Resources

User Management Tools

Content Management Tools

Filling System

Content Providers (ISVs)

Image source: developers.sun.com
MIDP OTA Specification

- Device Functionality
  » Support for HTTP 1.0
  » Discovery Application (to locate application and to download. Eg. Micro-browser)
  » AMS to manage OTA Application Provisioning life cycle. In MIDP it is called JAM-Java Application Manager

- OTA Application Provisioning life cycle (next slide)
MIDP OTA Specification
- OTA Application Provisioning life cycle

Discover module:
- In most cases it uses the HTTP or WAP micro browser. When the browser gets a MIDP application it sends it to the JAM to download and install.

Execute module:
- Allows user to select MIDlet suite and MIDlet.
- Starts the MIDlet in *Paused* state.
- Calls startApp() in the MIDlet to bring it to *Active* state

(Refer MIDlet Life cycle in previous slides.)
MIDP OTA Specification

- OTA Application Provisioning life cycle

(Installation and update module)
Removal Module: The application (complete MIDlet suite) and its associates RMS Entries will be removed. RMS – record store management system (local storage).
MIDP OTA Specification

Status Reports:

- 900 Success
- 901 Insufficient Memory
- 902 User Cancelled
- 903 Loss of Service
- 904 JAR size mismatch
- 907 Invalid JAR
- 909 Application authentication failure
- 910 Application authorization failure
- 912 Deletion Notification

Refer: [http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf](http://java.sun.com/products/midp/OTAProvisioning-1.0.pdf) for full list

**Provisioning portals** (download Servers) may take advantage of status reports to track the use of an application - for example, for billing purposes or to prioritize their content repository.
MIDP OTA Specification
Provisioning Portal (eg. J2EE based provisioning portal)

MIME types:
- JAD → text/vnd.sun.j2me.app-descriptor
- JAR → application/java-archive

Core packages:
- javax.provisioning
- javax.provisioning.adapter
- javax.provisioning.matcher

Further reading: http://developers.sun.com/techtopics/mobility/midp/articles/ota/