J2ME Programming

21W.780 – Class 2
February 13, 2007

Frank Bentley

Overview

What's the same?

What's missing?

JAD's and JAR's

Java Security

- Permissions
- Signing

Java Connector

- Networking
- File I/O

Messaging

RMS

Push Registry

Bluetooth

Location / Cell ID

GUIs

- LCD UI
- Game Canvas

What's the same?

At a high level, still java…

Everything inherits from java.lang.Object

Primitive types: int, float, boolean, byte, long, String

Basic util/lang classes: Vector, Hashtable, Calendar, Date, Thread, Timer

Basic i/o classes: InputStream, OutputStream, Reader, Writer, ByteArrayInputStream, ByteArrayOutputStream

What's Missing?

Most classes from after Java 1.1.8…

Collections: Set, TreeSet, Properties

XML: no built in XML support (can use KXmlParser)

File I/O: standard File I/O classes are different

Networking: standard networking classes are different

AWT/Swing: J2ME has its own GUI classes

High level APIs: SQL, JNI, etc.

JADs and JARs

All code and resources must be packaged in a JAR file

Metadata for the MIDlet goes into a JAD file

EclipseME takes care of generating basic JAD (file size, main MIDlet, etc.) and JAR

Example JAD:

MIDlet-Jar-URL: ZoneTag.jar
MIDlet-Jar-Size: 133285
MicroEdition-Configuration: CLDC-1.1
MIDlet-Version: 1.0.1
MIDlet-Name: ZoneTag
MIDlet-Vendor: Motorola
MicroEdition-Profile: MIDP-2.0
MIDlet-1: ZoneTag,,com.mot.labs.arc.zonetag.gui.ZoneTag
FlipInsensitive: True
MIDlet-Certificate-1-1:
MIIDojCCAoqgAwIBAgICCCwwDQYJKoZIhvcNAQEFBQAwfzELMAkGA1UEBhMCVVMxETAPBgNVBAgTCEIsbGlub2lzMRUwEwYDVQQHEwxMaWJlcnR5dmlsbGUxFTATBgNVBAoTDE1vdG9yb2xhIEluYzEMMAoGA1UECxMDUENTMSEwHwYDVQQDExhNYW51ZmFjdHVyZXIgRG9tYWluIDQwLTEwHhcNMDYxMTAzMDE0MTM5WhcNMjExMTAzMDE0MTM5WjA+MRUwEwYDVQQKEwxNb3Rvcm9sYSBJbmMxFTATBgNVBAsTDE1vdG9yb2xhIFBLSTEOMAwGA1UEAxMFQUM1NDEwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOxKOU3dXQgZqQzh++LbqVPi5n1fGPtKrL29xKejfwsOX5gX5rPDF//IyWZBs4ECrarptJvqZX8SLxmTwZUcQFz7y/2aOrF/FKYIzq/vlb/Y2gbtpdeOjkDmV0dcIdpa8zqc5fjopKXt5U+3/xbvyeX8d9aZUHFtGtpx28LpEd5zAgMBAAGjgewwgekwHwYDVR0jBBgwFoAUfHpHITdMaro5MMLC1vGdkWOiUy4wEQYJYIZIAYb4QgEBBAQDAgQQMA4GA1UdDwEB/wQEAwIF4DATBgNVHSUEDDAKBggrBgEFBQcDAzB/BghghkgBhvkBFQEB/wRwBo0ZYAgUCRZK1AAAAABV4AtQlhQJGxmFUEm2RClijMwMHMx0AEAAiABgIiKMLCwMnHQA0AAgAOAnZgYREQgHFq1vAAAAAGZgDTAZBwkbBChQQbZEIpwMLExsdIwAEACQAHAzrgbAEwYJJlPaAAAAADANBghghkgBhvkBFgQBATANBgkqhkiG9w0BAQUFAAOCAQEAI/iVlReYWELOHhRZqZBl8nv4hhkt0/xThydmIaX9Rp0scRDLr2DOGYnq0bqNGPljjw1jZFW/o3BUKTELK7+8YHwEURVitrSGlUFY/cQ8Qee+fgJJZBpJrfwJVmYEFfSRDUodRBypI+73Dj0PwYJVI5xWs2vqM6qSLK4c9xwwn1rwYJxs9aXKvrxO4ftqeIEZ8XYIMKpi+l7r02hBG9MMSpeCbbpe+1eSxF2DVXi/g/okY9lcVoxhko6paUZ3gZHTNEzAIPSU50/mGXRGl2ELdk2MOTPji04e/by1PKD8agJB/RpD3NaLzrcZsKxAGgGuoXqluMVRbgNFZoXa1IOh+A==
MIDlet-Jar-RSA-SHA1:
nhVNq8gI9K4vqr69YWOZXuz2mVtHOd7n3cOdeSXxd6XR0AMgTE3THLOVqZmQ+hAskdQ26/TlejyXHp6FrxrSRS7UlCFuoq8Ax6jaVqWBuo0WSR10hVFqDfvWXVrQO0l6Do0ZSYaPIFAXWOJzC0uUiclFDRBKdQd3WR2vSD/fv3s=
MIDlet-Permissions:
javax.microedition.io.Connector.http,javax.microedition.io.Connector.file.read,javax.microedition.io.Connector.file.write,javax.microedition.io.Connector.socket
MIDlet-Permissions-Opt: javax.microedition.media.control.VideoControl.getSnapshot

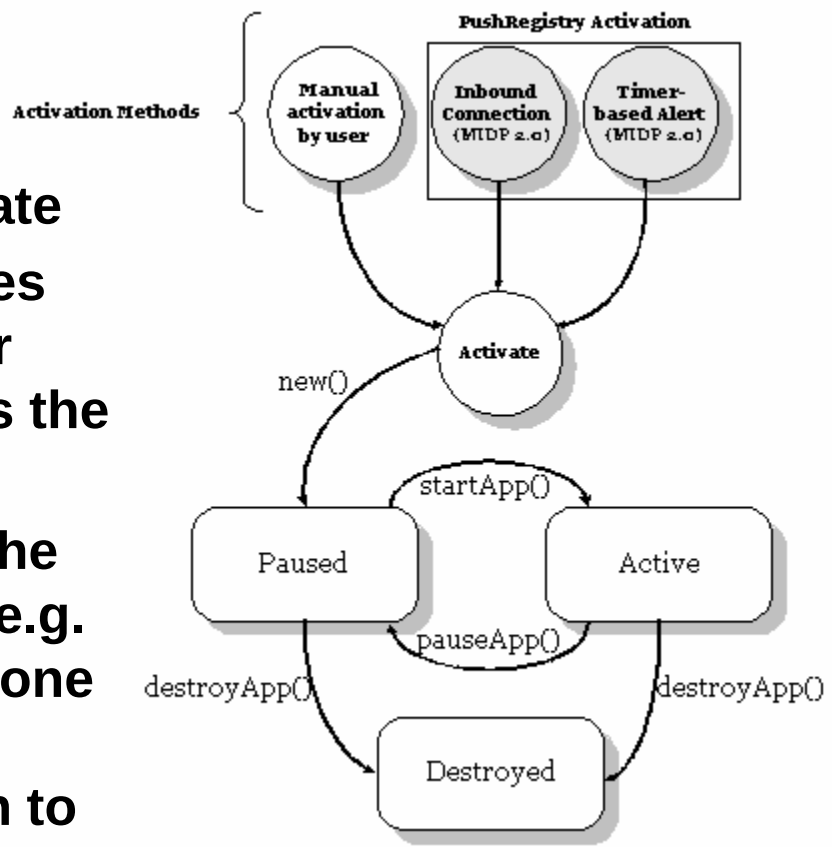
MIDlet lifecycle

Constructor – gets called once, when MIDlet is first created

startApp() – get called the first time the MIDlet is invoked and everytime the MIDlet resumes from a suspended state

pauseApp() – is called if the user changes focus to another phone application or phone event (e.g. phone call) requires the MIDlet to be suspended

destroyApp(boolean arg0) – is called if the user chooses to exit the application (e.g. presses red home button) or if the phone needs more resources for another process. You can throw an exception to say you'd really not like to quit, but the platform can override



From java.sun.com

Java Security - Permissions

Whenever you use a method that is protected, you must declare it in the permissions of the JAD file – example:

```
MIDlet-Permissions:
javax.microedition.io.Connector.http,javax.microedition.io.Connector
.file.read,javax.microedition.io.Connector.file.write,javax.microedi
tion.io.Connector.socket
```

Optional permissions allow application to run on phones that may not have all APIs - example:

```
MIDlet-Permissions-Opt:
javax.microedition.media.control.VideoControl.getSnapshot
```

If you don't declare a permission and try to use a protected API, likely a SecurityException will be thrown, or the KVM will just exit

Java Security - Permissions

Permissions for the v3x:

- javax.microedition.io.Connector.http
- javax.microedition.io.Connector.https
- javax.microedition.io.Connector.datagram
- javax.microedition.io.Connector.datagramreceiver
- javax.microedition.io.Connector.socket
- javax.microedition.io.Connector.serversocket
- javax.microedition.io.Connector.ssl
- javax.microedition.io.Connector.comm
- javax.microedition.io.PushRegistry
- javax.wireless.messaging.sms.send
- javax.wireless.messaging.sms.receive
- javax.microedition.io.Connector.sms
- javax.wireless.messaging.cbs.receive
- javax.microedition.media.control.RecordControl.record
- javax.microedition.media.control.VideoControl.getSnapshot
- javax.microedition.pim.ContactList.read
- javax.microedition.pim.ContactList.write
- javax.bluetooth

Java Security - Signing

Some APIs are protected such that only applications signed by the manufacturer or carrier can use them

Examples:

- Cell ID
- File I/O to parts of the phone file system
- Capturing a full-resolution image from the camera

Follow instructions in the Motorola certificate signing guide to use openssl to generate the SHA1 hash of your Jar file and add MIDlet-Certificate-1-1 and MIDlet-Jar-RSA-SHA1 to your JAD file before loading it onto the phone

Make sure you use the appropriate certificate for the IMEI of your phone

J2ME Connector API (1)

All stream I/O is initiated by the Connector class

Connector can get you HTTP streams, File streams, Sockets, SMS, etc.

HTTP Example:

```
StringBuffer s = new StringBuffer();
HttpConnection c =
      (HttpConnection)Connector.open("http://web.mit.edu/index.html");
InputStream is = c.openInputStream();
byte b;
while ((b = (byte)is.read()) != -1) {
   s.append((char)b);
}
is.close();
c.close();
```

J2ME Connector API (2)

File Example:

```
FileConnection sc = (FileConnection)Connector.open("file:///c/mobile/picture/tmp.txt");
OutputStream os = sc.openOutputStream();
os.write(("text to go into the file").getBytes());
os.flush();
os.close();
```

Don’t forget to add the appropriate permissions to your JAD file and sign if necessary (i.e. when using files)!!

Messaging

You can send and receive SMS messages from Java. To talk between applications, you can address an SMS to a specific port that an application on another phone can listen to

Can send/receive both text and binary messages (limited to 160 bytes)

Example:

```
sender = (MessageConnection)Connector.open("sms://+16172531000:9532 ");
TextMessage t = (TextMessage)sender.newMessage(MessageConnection.TEXT_MESSAGE);
t.setPayloadText(“Hello World”);
t.setAddress(“sms://+16172531000:9532”);
sender.send(t);
```

Server:

```
serverConn = (MessageConnection)Connector.open ("sms://:9532");
serverConn.setMessageListener(this); // where this implements MessageListener
```

RMS

RMS – the Record Management Store – is an easy place to store persistent data

Can create multiple stores that each contain a set of records

Example adding a record:

```
RecordStore rs = RecordStore.openRecordStore("MyAppointments",true);
String appt = "new record";
byte bytes[] = appt.getBytes();
rs.addRecord(bytes,0,bytes.length);
rs.closeRecordStore();
```

Example reading a record:

```
RecordEnumeration re = rs.enumerateRecords(null, null, false);
     if (re.hasNextElement())
        byte nextRec[] = re.nextRecord();
```

More info: http://www-128.ibm.com/developerworks/library/wi-rms/

Push Registry

The push registry is a way to automatically start a MIDlet after a specified amount of time or when a given system event occurs (e.g. incoming SMS)

You can register statically in the JAD, or dynamically in the constructor of your MIDlet…

Example:

MIDlet-1: PushMIDlet,,j2medeveloper.basicpush.PushMIDlet

MIDlet-2: WMAMIDlet,,j2medeveloper.wma.WMAMIDlet MIDlet

Name: MyMIDletSuite MIDlet-Vendor: Sun Microsystems, Inc.

MIDlet-Version: 1.0 MIDlet-Jar-Size: 4735

MIDlet-Jar-URL: basicpush.jar

MicroEdition-Configuration: CLDC-1.0

MicroEdition-Profile: MIDP-1.0

MIDlet-Push-1: socket://:5000, j2medeveloper.basicpush.PushMIDlet, *

MIDlet-Permissions: javax.microedition.io.PushRegistry, javax.microedition.io.Connector.serversocket

More info: http://developers.sun.com/techtopics/mobility/midp/articles/pushreg/

Bluetooth (1)

Implements device discovery, obex, and serial communication

DiscoveryAgent handles discovering proximate devices

```
LocalDevice localDevice = LocalDevice.getLocalDevice();
discoveryAgent = localDevice.getDiscoveryAgent();
discoveryAgent.startInquiry(accessCode,
   aDiscoveryListener);
```

DiscoveryListener.deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) gets called each time a device is found

For OBEX example see:

http://developers.sun.com/techtopics/mobility/apis/articles/bluetoothobex/

Bluetooth (2)

Serial (RFCOMM) API just uses the Connector API

```
// assuming the service UID has been retrieved
String serviceURL =
    "btspp://localhost:"+serviceUID.toString());
try {
    // create a server connection
    StreamConnectionNotifier notifier =
       (StreamConnectionNotifier) Connector.open(serviceURL);
    // accept client connections
    StreamConnection connection = notifier.acceptAndOpen();
    // prepare to send/receive data
    byte buffer[] = new byte[100];
    String msg = "hello there, client";
    InputStream is = connection.openInputStream();
    OutputStream os = connection.openOutputStream();
    // send data to the client
    os.write(msg.getBytes());
    // read data from client
    is.read(buffer);
    connection.close();
} catch(IOException e) {
  e.printStackTrace();
}
```

More Info: http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/
http://developer.motorola.com/docstools/technicalarticles/Bluetooth_20060601.pdf

Location

You can get the current Cell ID through java system properties

The combination of cell id, lac, mnc, and mmc form a globally unique ID

Systems like ZoneTag can map a cell ID into a location, or you can have the user provide semantic labels (e.g. PlaceIts)

```
String cellID = System.getProperty("CellID");
String lac = System.getProperty("LocAreaCode");
String imsi = System.getProperty("IMSI");
String mcc = imsi.substring(0,3);
String mnc = imsi.substring(3,6);
```

More on location on 3/6

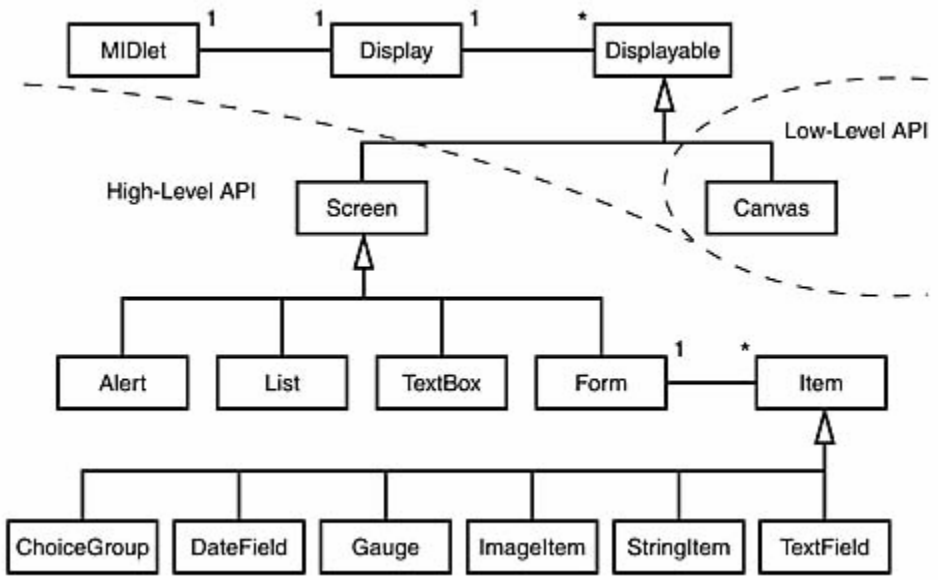
GUIs

Two different ways to build a screen in J2ME:

- LCDUI
  - Standard widgets (lists, checkboxes, text fields, etc.)
  - Look like standard platform components on any phone platform
  - Easy to implement
  - Don’t allow access to number pad except when entering text
- Game Canvas
  - Can implement your own widgets
  - Full control of screen and keypad
  - More difficult (e.g. what happens to a widget on a different screen or a phone with different buttons)

LCD UI

- LCD UIs are a series of screens
- Common interactions are wizard-like with a series of lists as the interaction
- Forms can contain a series of items on one screen
- Can intermix LCD UI screens and canvas-based screens in an application



from http://www.developer.com/java/j2me/article.php/10934_1561591_1

Canvas screens

Canvas screens are the lowest level API to the screen in J2ME

Must implement paint() method to draw components to the screen

Much like low-level Graphics API in desktop java

Next Steps…

Due next week:

- **Project proposal presentation in class**
- **Written proposal due 3/1 (Ed's office) – one per group**

Due 3/6:

- **Build a simple phone application that uses networking or messaging APIs**