CS 683 Emerging Technologies
Fall Semester, 2004
Doc 18 J2ME Intro
Contents

References

J2ME in a Nutshell, Kim Topley, O'Reilly, 2002

Sun's J2ME Wireless Toolkit Documentation

http://java.sun.com/j2me/docs/index.html

Documentation in J2ME Wireless Toolkit download

http://www.eli.sdsu.edu/courses/fall04/cs683/j2me/index.html

J2ME Wireless Toolkit 2.1

Download at:

http://java.sun.com/products/j2mewtoolkit/download-2_1.html

Installation instructions are on the above page

Documentation

User's Guide

http://java.sun.com/j2me/docs/wtk2.1/user_html/index.html

Also included in Wireless Toolkit download (docs directory)

Sun's J2ME Documentation Page

http://java.sun.com/j2me/docs/index.html

J2ME API Documentation

Included in the WTK (Wireless Toolkit)

For the course J2ME Wireless Toolkit documentation:

http://www.eli.sdsu.edu/courses/fall04/cs683/j2me/index.html

J2ME Overview

- Configurations

  Base configuration for a range of devices

  Connected Limited Device Configuration (CLDC)

  Connected Device Configuration (CDC)

- Profiles

  Additions to configurations for a particular device

Connected Limited Device Configuration (CLDC)

Basic cell phone & PDA

- A 16-bit or 32-bit processor clock speed of 16MHz or higher
- At least 160 KB of non-volatile memory
  For CLDC libraries and virtual machine
- At least 192 KB of total memory available
  For Java platform
- Low power consumption, often operating on battery power
- Connectivity to some kind of network, often with a wireless, intermittent connection and limited bandwidth

Connected Device Configuration (CDC)

High-end PDAs, set-top boxes

- 32-bit microprocessor/controller
- 2MB of RAM for Java
- 2.5 MB of ROM Java

Other Java/J2ME Technologies

- Java Card
  - For smart cards
- Java Telephone API (JTAPI)
  - Call centers
- Java Embedded Server
  - Applications & services on embedded devices
  - 1MB persistent storage + run-time cache
- PersonalJava
  - Obsolete
  - Replaced by CLDC & CDC
- Java TV API
  - Digital television receivers
- JavaPhone API
  - Direct telephony control
  - Datagram messaging
  - Address book and calendar information
  - User profile access
  - Power monitoring
  - Application installation

Connected Limited Device Configuration (CLDC) Profiles & API

Mobile Information Device Profile (MIDP)

- Basic Profile for Cell phones & PDAs
- Versions 1.0 & 2.0
- Included in J2ME Wireless Toolkit 2.1
- MIDP for Palm OS

  http://java.sun.com/products/midp4palm/

  Based on MIDP 1.0

Wireless Messaging API (WMA)

Platform-independent access to wireless communication resources like Short Message Service (SMS)

- Included in J2ME Wireless Toolkit 2.1

Connected Limited Device Configuration (CLDC) Profiles & API

Mobile Media API (MMAPI)

- Audio
- Video
- Time-based multimedia support
- Included in J2ME Wireless Toolkit 2.1

Information Module Profile (IMP)

- Emergency call boxes
- Parking meters
- Wireless modules in home alarm systems
- Industrial metering devices

Location API

- Addresses mobile positioning

SIP API

- Targets mobile phones

- Establish and manage multimedia IP sessions

Connected Limited Device Configuration (CLDC) Profiles & API

Security and Trust Services API (SATSA)

- Extends the security features ofJ2ME
- Cryptographic APIs
- Digital signature service
- User credential management

Mobile 3D Graphics API

Scalable, small-footprint, interactive 3D API for use on mobile devices

J2ME Web Services APIs (WSA)

Web services for J2ME

Bluetooth API

By Motorola/Freescale
Status unknown

Connected Limited Device Configuration (CLDC)

Java Limitations

- No Floating point

  No float or double
  No method that uses float or double

- No Reflection
- No Weak References
- No Object finalization
- No daemon threads
- Fewer Exception classes
- No Java native interface

Security

J2SE security model not used

Code runs in a sandbox

Class Verification

Must run byte-code verifier by hand before running application

Packages from J2SE

java.io

ByteArrayInputStream	ByteArrayOutputStream
DataInputStream	DataOutputStream
InputStream	InputStreamReader
OutputStream	OutputStreamWriter
PrintStream	Reader
Writer	

java.lang

Boolean	Byte	Character
Class	Integer	Long
Math	Object	Runtime
Short	String	StringBuffer
System	Thread	Throwable

java.util

Calendar	Date	Hashtable
Random	Stack	Timer
TimerTask	TimeZone	Vector

Some classes are based on JDK 1.1.8 or earlier

Some classes are smaller than their J2SE equivalents

HelloWorld Example

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

public class MySample extends MIDlet {

   public MySample() {
   }

   public void startApp() {
      Form form = new Form( "First Program" );
      form.append( "Hello World"  );
      Display.getDisplay(this).setCurrent( form );
   }

   public void pauseApp() {
   }

   public void destroyApp( boolean unconditional ) {
   }

}
```

Running the Example

You must download and install the J2ME Wireless Toolkit 2.1

Download at:

http://java.sun.com/products/j2mewtoolkit/download-2_1.html

Installation instructions are on the above page

Example will use Unix path conventions

Assume that the Wireless Toolkit (WTK) is installed in

/pathtoWTK/WTK2.1

Running the Example the Hard Way

Create a directory, MySample, for your project

Create subdirectories:

- classes
- src
- tmpclasses

Place the source in file MySample.java and place file in directory src

In the directory MySample compile using:

```
javac -d tmpclasses -bootclasspath
   /pathtoWTK/WTK2.1/lib/cldcapi10.jar:/pathtoWTK/WTK2.1/lib/
midpapi20.jar
   -classpath tmpclassese:classes src/*javapreverify -classpath
```

Then run

```
/pathtoWTK/WTK2.1/bin/preverify -classpath
   /pathtoWTK/WTK2.1/lib/cldcapi10.jar:/pathtoWTK/WTK2.1/lib/
midpapi20.jar
   -d classes tmpclasses
```

Running the Example the Hard Way Continued

Create the file

MANIFEST.MF

```
MIDlet-1: MySample, MySample.png, MySample
MIDlet-Name: MySample
MIDlet-Vendor: Unknown
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-2.0
```

Create the jar file using

```
jar cfm MySample.jar MANIFEST.MF -C classes .
```

Create the file

MySample.jad

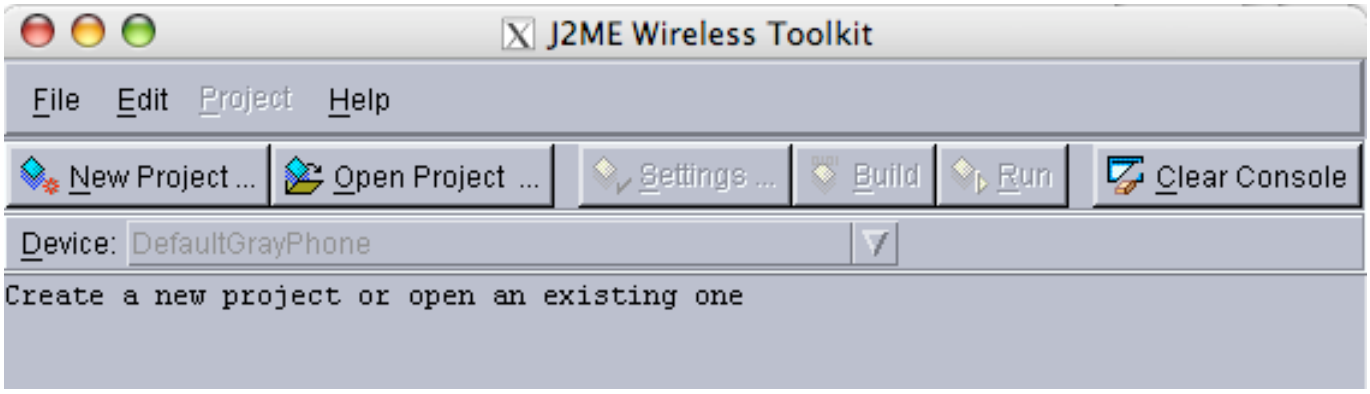
```
MIDlet-1: MySample, MySample.png, MySample
MIDlet-Jar-Size: 1433
MIDlet-Jar-URL: MySample.jar
MIDlet-Name: MySample
MIDlet-Vendor: Roger Whitney
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-2.0
```

Run the emulator

Running the Example the Easy Way

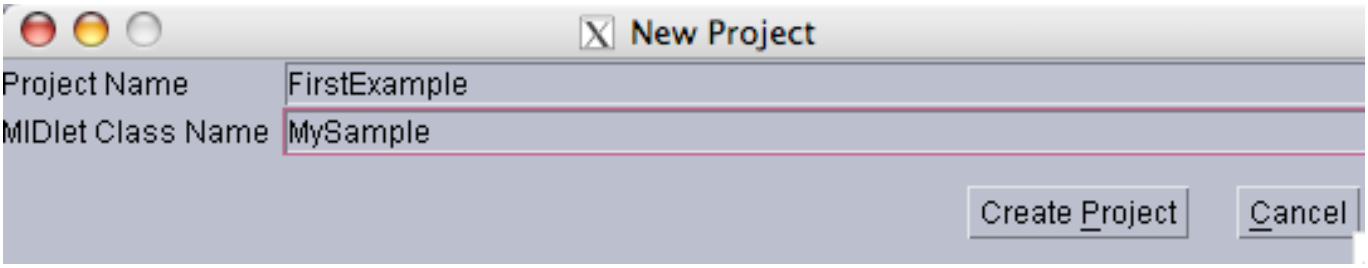
Run the program

/pathtoWTK/WTK2.1/bin/ktoolbar



Click on New Project.

Enter the Project name and class name as shown below



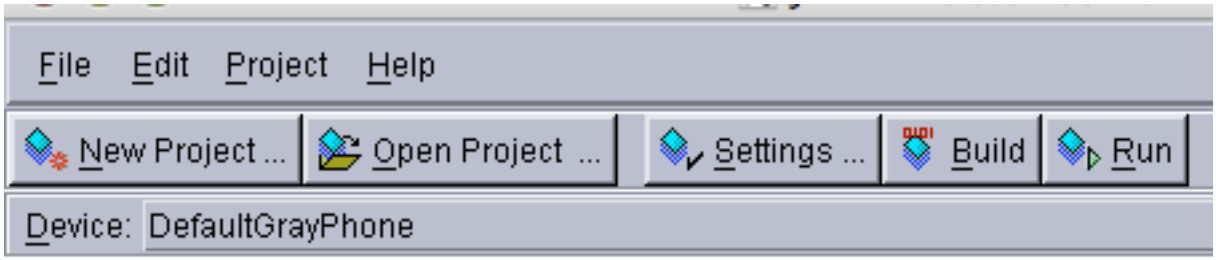
Click on Create Project. A setting window will open up. Accept the defaults by clicking ok in that window.

Running the Example the Easy Way Continued

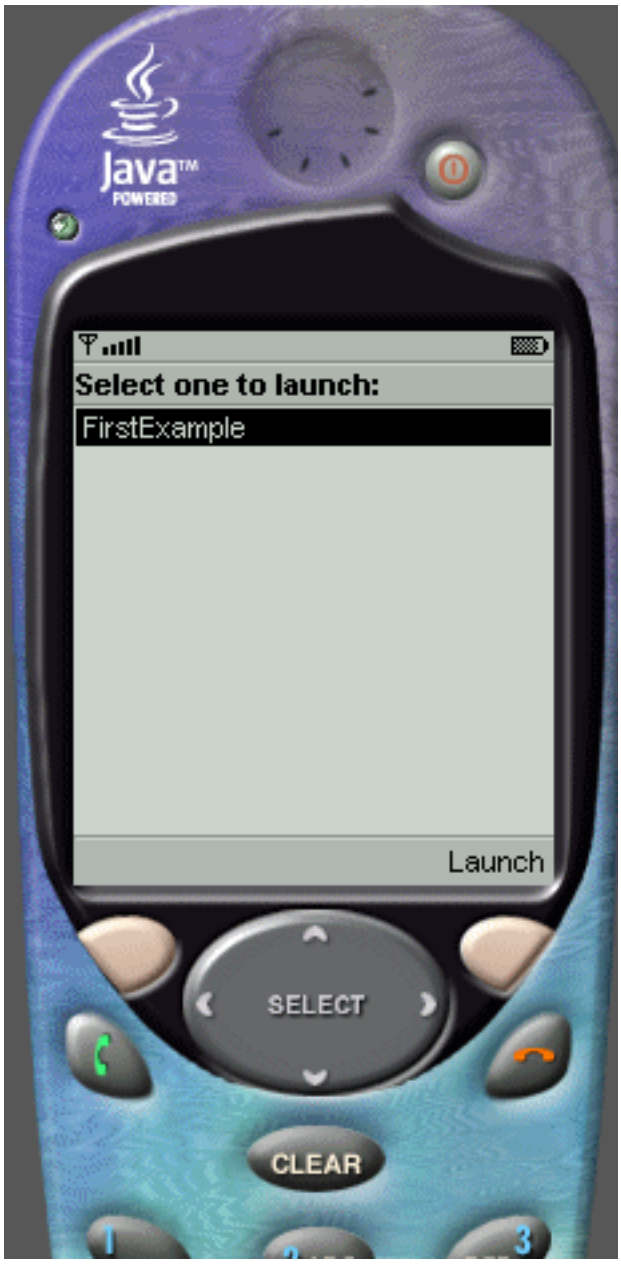
Place the source code for the class in the file

WTK2.1/apps/FirstExample/src/MySample.java

In the ktoolbar main window click on the “Build” button. When the build compiles successfully then click on the “Run” button.



The emulator then opens up and one gets a window showing a cell phone



javax.microedition.midlet.MIDlet & MIDlet States

MIDlet States

- Active
- Paused
- Destroyed

When loaded MIDlet

Starts in the Paused state
Normal instance initialization is done
If constructor throws an exception the MIDlet is destroyed

MIDlet Methods

startApp()

```
protected abstract void startApp()
    throws MIDletStateChangeException
```

Called when MIDlet becomes active

If a transient failures occurs:

- Thrown the MIDletStateChangeException exception
- MIDlet is moved to Paused state

If a non-transient failures

- StartApp() should call `notifyDestroyed`()

If any exception other than MIDletStateChangeException
- The MIDlet is destroyed

startApp() Verse Constructor

MIDlet can have a no argument constructor

Use constructor to create resources once

Use startApp() for resources that need action each time move to active state

Two Programs with the Same Behavior

```
public class MySample extends MIDlet {

   public MySample() { }

   public void startApp() {
      Form form = new Form( "First Program" );
      form.append( "Hello World"  );
      Display.getDisplay(this).setCurrent( form );
   }

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

```
public class MySample extends MIDlet {

   public MySample() {
      Form form = new Form( "First Program" );
      form.append( "Hello World"  );
      Display.getDisplay(this).setCurrent( form );
 }

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

pauseApp()

protected abstract void **pauseApp**()

Called when MIDlet is moved to Paused state from Active state

If a runtime exception occurs in pauseApp()
- MIDlet will be destroyed,
- destroyApp() will be called

destroyApp()

protected abstract void destroyApp(boolean unconditional) throws MIDletStateChangeException

Called when MIDlet is to enter the Destroyed state

Should
- Release all resources
- Save any persistent state

If unconditional is false MIDlet can

throw MIDletStateChangeException to signal not to destroy it

notifyDestroyed()

public final void notifyDestroyed()

MIDlet uses to notify it has entered the Destroyed state

destroyApp() will not be called

MIDlet must perform same operations as done by destroyApp()

notifyPaused()

public final void notifyPaused()

MIDlet uses to notify it has entered the Paused state

Application has to call resumeRequest() to reenter the Active state

resumeRequest()

public final void **resumeRequest**()

getAppProperty()

public final String getAppProperty(String key)

Returns the value of the property key

checkPermission()

public final int checkPermission(String permission)

Added MIDP 2.0

Returns the value of the given permission

0 permission denied
1 permission granted

platformRequest()

```
public final boolean platformRequest(String URL)
  throws ConnectionNotFoundException
```

Added MIDP 2.0

Requests that device handle the indicated URL

URL can be a

- Jar to load or install
- tel:<number> a phone number to call

Using System.out

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

public class MySample extends MIDlet {

   public MySample() {
      Form form = new Form( "First Program" );
      form.append( "Hello World"  );
      Display.getDisplay(this).setCurrent( form );

   }

   public void startApp() {
      System.out.println( “Start”);
   }

   public void pauseApp() {
      System.out.println( “Pause”);
   }

   public void destroyApp( boolean unconditional ) {
      System.out.println( “Good bye”);
   }
}
```

System.out can be used for debugging. When run in the simulator, the output is put in the console, not the phone