Praktikum Entwicklung von Mediensystemen
Mobile Dienste für Studenten

Http and Record Stores – 10/30/2009
Outline:

- Solution for exercise 1
  - Camera MIDlet
  - Audio MIDlet
- Mobile Http-connections and Record Stores
- Exercise 2
Camera MIDlet

1. Create player and control:

```java
//initialize player with camera source
player = Manager.createPlayer("capture://video");

//realize player
player.realize();

// VideoControl object for controlling the display
videoControl = (VideoControl)player.getControl("VideoControl");
```
Camera MIDlet

2. Show video source on screen (1/2):
   • Attach to Form

   ```java
   if(videoControl!=null) {
       Item videoItem = (Item)(videoControl.initDisplayMode(
           VideoControl.USE_GUI_PRIMITIVE,null));
       videoItem.setLayout(Item.LAYOUT_CENTER |
           Item.LAYOUT_NEWLINE_AFTER);
       form.append(videoItem);
   }
   ```
Camera MIDlet

2. Show video source on screen (2/2):
   - Attach to Canvas

   ```
   int width = getWidth();
   int height = getHeight();

   videoControl.initDisplayMode(
       VideoControl.USE_DIRECT_VIDEO, this);
   try { // display video 2 pixels away from borders
       videoControl.setDisplayLocation(2, 2);
       videoControl.setDisplaySize(width - 4, height - 4);
   }...
   videoControl.setVisible(true);
   ```
Camera MIDlet

3. Start Player:

```java
// starts the player
// prefetch() is called implicitly
player.start();
```
Camera MIDlet

4. Make a snapshot (1/2):

- Only a snapshot of the current video stream.
- Java ME cannot use the full available camera resolution.
  => must be considered at design time
- e.g. Nokia 3650 supports 160 x 120 pixels only
- Use getSnapshot() method with a supported image type or null for standard encoding (mostly PNG)
Camera MIDlet

4. Make a snapshot (2/2):

```java
new Thread() { // anonymous Thread definition
    public void run() {
        try {
            // synchronize this
            synchronized(CameraMidlet.this) {
                byte[] imageData;
                imageData = videoControl.getSnapshot(null);
            }
        } catch(MediaException e) {...}
    }
}.start();
```
Camera MIDlet

5. Close Player

- Close the Player at the end
- e.g. when exiting the MIDlet

```java
player.close();
```
Camera MIDlet
6. Show Picture:

```java
// create image out of the byte array
Image photo = Image.createImage(imageData, 0,
    imageData.length);
// append the image directly to a form
form.append(photo);

or

// create an ImageItem and append this
ImageItem photoItem = new ImageItem(null,
    photo, ImageItem.LAYOUT_CENTER, null);
form.append(photoItem);
```
Audio MIDlet

1. Create player and control:

```java
// create the player using standard encoding
player = Manager.createPlayer("capture://audio");
player.realize();
rc = (RecordControl)player.getControl("RecordControl");

// create an OutputStream for the RecordControl
output = new ByteArrayOutputStream();
rc.setRecordStream(output);
```
Audio MIDlet

2. Start Recording:
   - Recording should run in an extra Thread
   - Will run until stopped

   // At first start the control
   rc.startRecord();

   // then start the player
   player.start();
Audio MIDlet

3. Stop Recording:

    // committing the control finishes recording
    rc.commit();

    // save the recordedData in a byte array
    byte[] soundData = output.toByteArray();

    // close the player
    player.close();
Audio MIDlet

4. Play recorded data:

```java
// create an input stream with the byte array
ByteArrayInputStream audioInputStream = new ByteArrayInputStream(soundData);
player = Manager.createPlayer(audioInputStream, "audio/x-wav");
player.prefetch();

// add PlayerListener for checking on player status
player.addPlayerListener(this);
player.start();
```
Hints and Tips:

- Packages should be used but in a reasonable number (virtual machine might have problems with too many packages)

- On device debugging is terrible but some tools like MyRedirector (http://www.mobile-j.de/snipsnap/space/J2ME/System.out+redirect+on+S60+3rd+Edition) are helpful (pre-installed on many mobile phones in our labs)
Generic Connection Framework (GCF) – part of CLDC – is the basis for network programming in Java ME

Collection of interfaces in CLDC - implemented by MIDP

No constructors for various connection objects – constructed by calling static method `open(url)` of factory-class `Connector`

Returns an object that implements one of the generic connection interfaces which is specified by a protocol-identifier

CLDC defines interfaces for HTTP, socket, datagram and serial port

MIDP additionally supports HTTPS, server-socket and SSL

```java
HttpConnection hc =
    (HttpConnection)Connector.open("http://www.someurl.de");

Connector.open("socket://127.0.0.1:32780");
Connector.open("comm:IR0;baudrate=19200");
Connector.open("file:/myFile.txt");
```
Persistent Storage: Record Stores

- Record store
  - contains records (pieces of data)
  - instance of javax.microedition.rms.RecordStore

- Every MIDlet in a MIDlet Suite can access every Record Store

- Since MIDP 2.0:
  - Access across Suite boarders possible !!!
Java ME provides only limited features to read and write data on mobile devices and store them permanently.

- Package `javax.microedition.rms`; main class `RecordStore` => named „database“; collection of uniquely identified Records (byte arrays).
- `RecordStores` are identified by their name, Records by their ID (primary key).
- RMS allows manipulation and sharing of records within `RecordStores`.
- Accessing records by their ID is often tedious, use `Enumeration` instead.

```java
RecordStore db = RecordStore.openRecordStore("myDBfile", true);
byte[] b = baos.toByteArray();
myDBfile.addRecord(b, 0, b.length);
```
Goal:
- Introduction to Java ME network programming and Record Management System (RMS)
- Part of CLDC and MIDP => should run on all mobile phones

Task:
- Create a MIDlet that sends an arbitrary string to a server and displays the answer
- Implement a history-feature that stores the last 5 answers from the server permanently – even after the MIDlet is shut down

Server URL:
Different Tasks/Interfaces:
- Inserting an arbitrary string
- Checking the validity of the input (no input – no connection)
- Sending the input to the server (e.g. display a waiting screen)
- Displaying the answer from the server
- Displaying a history of the last 5 answers (which are stored permanently)

Connection Setup/Threading:
- Use either HTTP POST- or GET-request
- Run Http-connection in extra thread
- Simple threading enough
Solution:

- Must work on a mobile phone (test on emulator first)
- Mobile phones and SIM cards can be lent from the university (Alexander De Luca, 5th floor)

Deadline and Submission:

- Each student must submit his/her own solution via email to gregor.broll(at)ifi.lmu.de and alexander.de.luca(at)ifi.lmu.de by Thursday, November 5rd, 12 p.m.
- Create a zip-file named after you and insert a folder called excercise2 containing your solution.
Questions?

Have fun!