# Mensch-Maschine-Interaktion 2

## Übung 5

(12./14./15. Juni 2007)

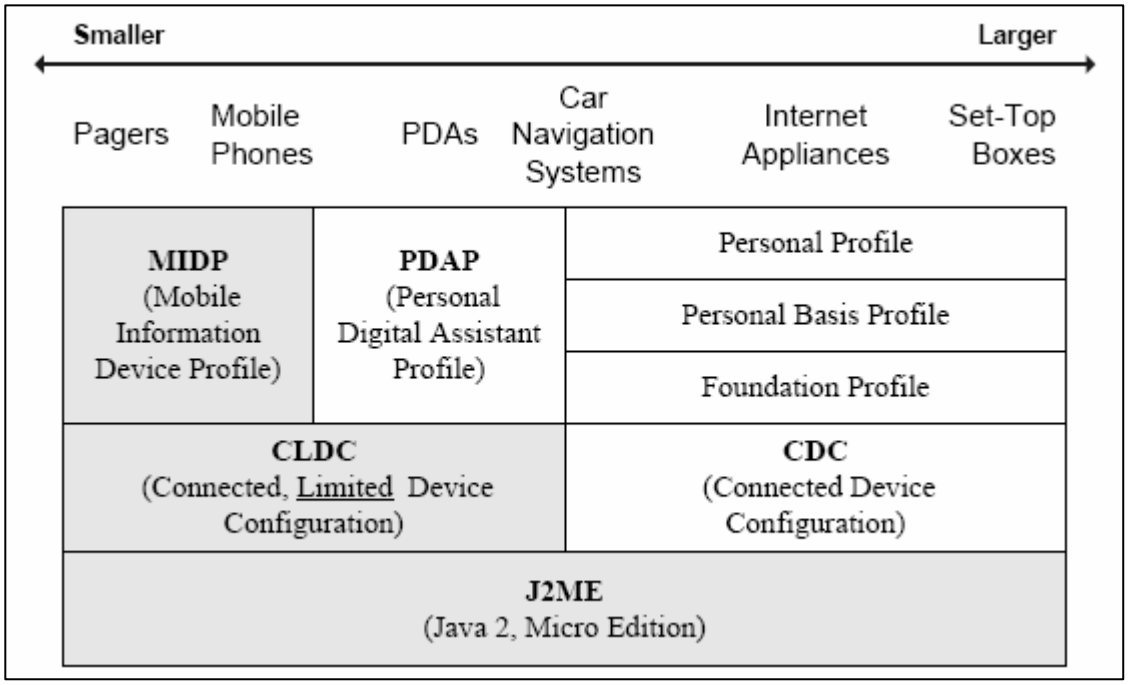
Arnd Vitzthum - arnd.vitzthum@ifi.lmu.de
Amalienstr. 17, Raum 501
Dominic Bremer - bremer@cip.ifi.lmu.de

# Java ME – Overview (I)

- Java ME
  - slim Java for mobile devices
  - Java ME stack
    - configuration + profile + additional APIs
  - Configuration
    - JVM + minimal amount of functionality
    - subset of Java SE
    - e.g. CLDC 1.1
  - Profiles
    - enhance the configuration with functionality
    - APIs for user interface, persistent storage, etc.
    - e.g. MIDP 2.0
  - Additional APIs for Bluetooth connections, Multimedia and more

# Java ME – Overview (II)

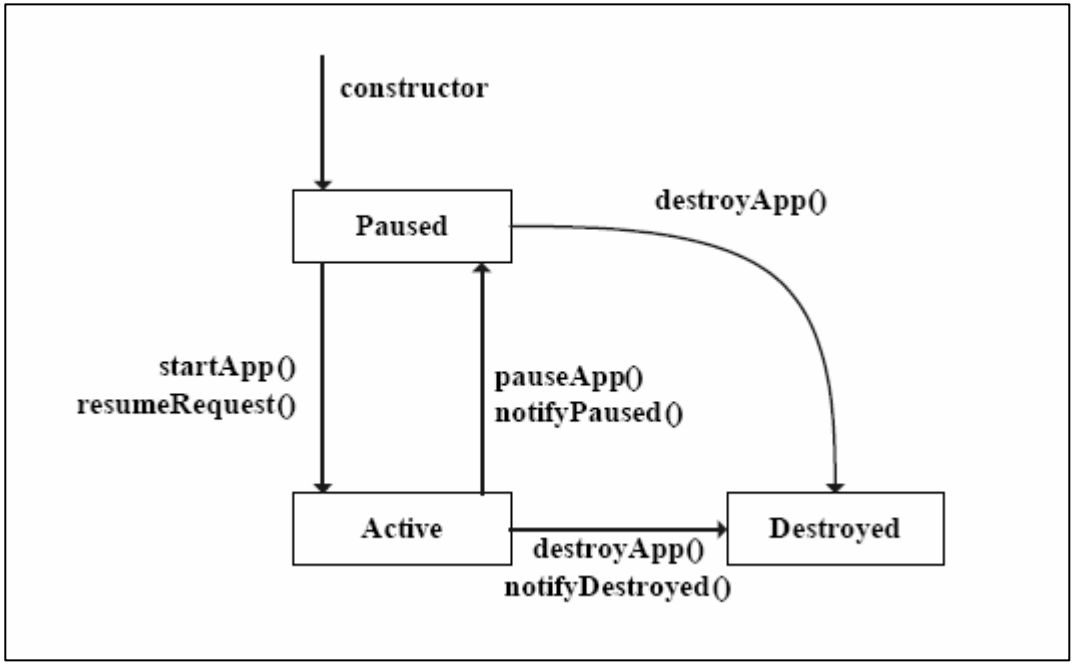
- The Java ME Universe



# MIDlets

- MIDP applications are called MIDlets
- Every MIDlet is an instance of javax.microedition.midlet.MIDlet
  - Constructor
  - Implements lifecycle methods
- Conceptually similar to Applets
  - Executed in host environment

# MIDlet Life Cycle (I)



# MIDlet Life Cycle (II)

- Application Manager (also called Application Management Software - AMS) controls the installation and execution of MIDlets
- Start of a MIDlet: constructor + startApp() are executed by the Application Manager
- MIDlet
  - notifies AMS that it has entered into *Paused* state (*notifyPaused()*)
  - notifies AMS that it has entered into *Destroyed* state (*notifyDestroyed()*)

# MIDlet Build Cycle (I)

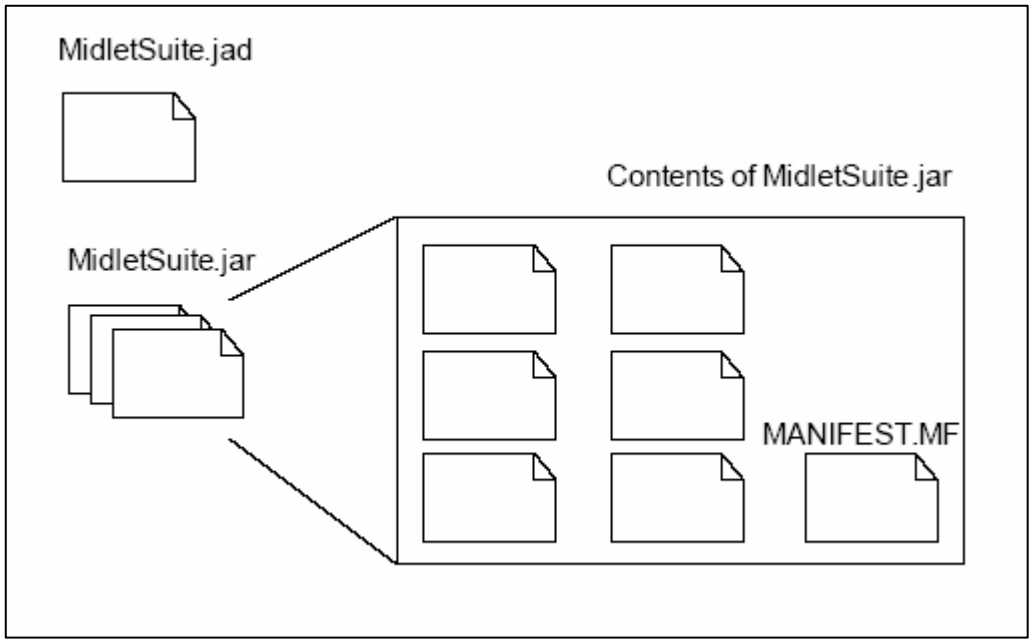
1. Edit source code
2. Compile (like compiling normal java)
3. Preverify
   - Bytecode verification (makes sure it behaves well + won't do nasty things) is split into two steps
   - lightweight second verification on the mobile device (standard verification too memory intensive)
   - special class format (adds 5% to normal class file size)
   - Normally not visible for the programmer

# MIDlet Build Cycle (II)

4. (Application) Package, MIDlet Suite
   - MIDlets + Classes + Resources + Manifest Information => Java Archive (JAR)
   - Manifest: describes content of archive (versions of CLDC and MIDP, name, version, vendor)
   - Application Descriptor (*.jad)
     - Same information like manifest (+ MIDlet-Jar-Size, MIDlet-Jar-URL), but a external file
     - Normally used for installation
5. Test or Deploy

# MIDlet Suite

- Anatomy of a MIDlet Suite



# MIDP: User Interface

- Goal: Write Once, Run Anywhere
- Anywhere?
  - Different screen sizes
  - Resolution of screen
  - Color or grayscale screen
  - Different input capabilities (numeric keypad, alphabetical keyboards, soft keys, touch screens, etc.)
- Abstraction required
  - specifying a user interface in abstract terms
  - (Not:) “Display the word ‘Next’ on the screen above the soft button.”
  - Rather: “Give me a ‘Next’ command somewhere in this interface”

# Example – Form

```
Form form2 = new Form(„Text anzeigen");
form2.append(„Hello World“);
form2.addCommand(new Command("Zurück", Command.BACK, 1));
form2.setCommandListener(new CommandListener() {
        public void commandAction(Command c, Displayable d) {
                …
            //back to the main menu
            showMenu();
        }
});
…
Display.getDisplay(this).setCurrent(form2);
…
private void showMenu() {…}
```

# Persistent Storage (I)

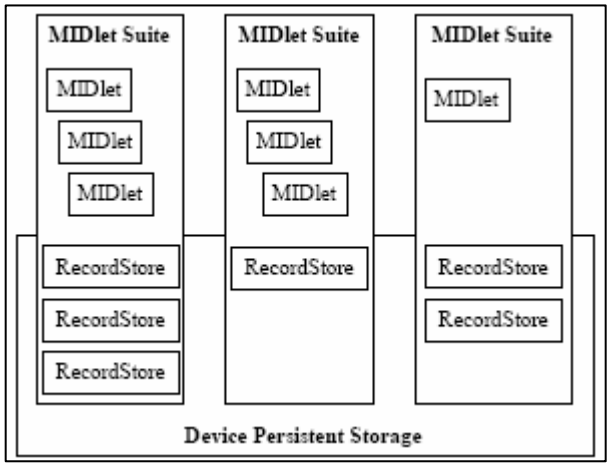
- Goal: Write Once, Run Anywhere
- Anywhere?
  - Device with Flash ROM
  - Battery-backed RAM
  - Small Hard Disk

=> Abstraction is needed

- Record stores (small databases)
- Min. 8KByte (Nokia 6600: ‘the only limitation is the amount of free memory’)

# Persistent Storage (II) – 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 !!!



# Example – Record Store

- Open record store:

```
RecordStore db = RecordStore.openRecordStore("myDBfile", true);
```

- Close record store:

```
db.closeRecordStore();
```

- Write string *record:*

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
DataOutputStream dos = new DataOutputStream(baos);
dos.writeUTF(record);
byte[] b = baos.toByteArray();
db.addRecord(b, 0, b.length);
```

- Read record with index *i*:

```
byte[] record = db.getRecord(i);
ByteArrayInputStream bais = new ByteArrayInputStream(record);
DataInputStream dis = new DataInputStream(bais);
String in = dis.readUTF();
```

# Recommended IDEs

- Netbeans (http://www.netbeans.org/products/ide/) + Mobility Pack (http://www.netbeans.org/products/mobility/)
  - Much better Java ME support then EclipseME (e.g. graphical interface editor)
- Eclipse (http://www.eclipse.org/) + EclipseME (http://eclipseme.org/docs/index.html)
  - May be better for developers experienced in using eclipse