Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Outline

History
J2ME Basics
Acronyms: CDC/CLDC/MIDP
Midlets
Developing a user interface / storing data
Resources / Documents / Tools (IDEs)
Implementing “Hello World”
Experiences

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Java on mobile devices: History [4,1]

- 1990: Java started as an internal project at Sun Microsystems
- 1996: Initial release of JDK 1.0 (applets → servlets)
- 1999: JavaOne conference
  - Subdivision of Java in
    - Java 2 Enterprise Edition (J2EE)
    - Java 2 Standard Edition (J2SE)
    - Java 2 Micro Edition (J2ME) (successor of Personal Java and Embedded Java)
- 2000/01 First mobile phones with support for J2ME

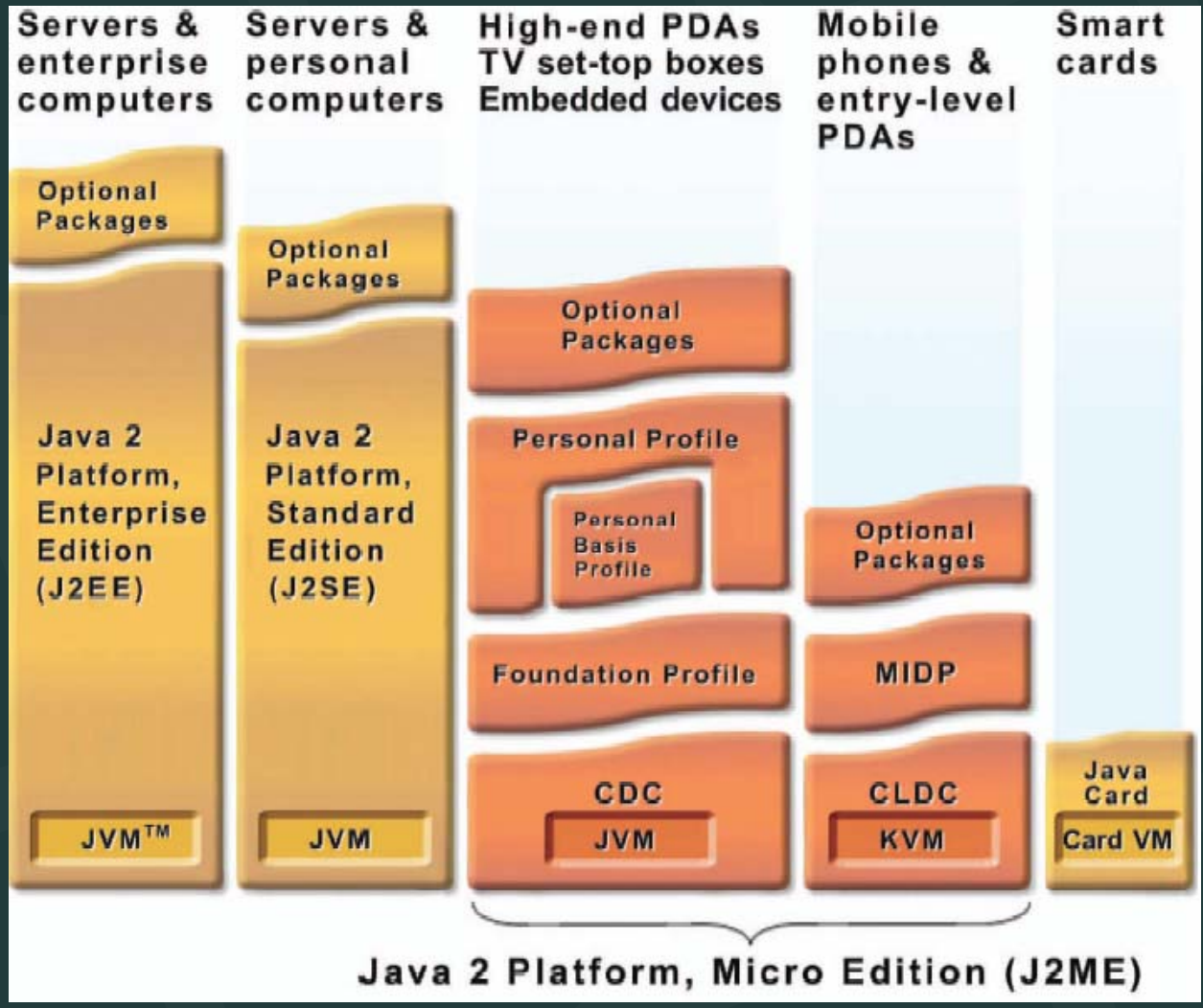
Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Java on mobile devices: History

- 2002: Second version of Mobile Information Device Profile (MIDP 2.0)
- April 2004: 250 Million mobile phones support J2ME
- October 2004: 1700 Million mobile phones support J2ME
- Now: most vendors of mobile phones (Motorola, Nokia, Panasonic, Samsung, Sharp, Siemens, Sony Ericsson, Toshiba, etc.) provide mobile phones that support J2ME

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

The Java universe



Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: Basics

J2ME: Java 2 Platform, Micro Edition
- "Java for small devices"
- Divided in configurations, profiles and optional APIs

Stack
- Configuration + profile + optional APIs

Configuration: for a specific kind of device
- Specifies a Java Virtual Machine (JVM)
- Subset of J2SE (Standard Edition)
- Additional APIs

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: Basics

Profile: more specific than configuration

- based on a configuration
- adds APIs for user interface, persistent storage, etc.

Optional APIs: additional functionality (Bluetooth, Multimedia, Mobile 3D, etc.)

Everything is specified by a JSR (Java Specification Requests)

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

The J2ME universe

Smaller ⟷ **Larger**

Pagers | Mobile Phones | PDAs | Car Navigation Systems | Internet Appliances | Set-Top Boxes

<table>
<tr><td rowspan="2">MIDP
(Mobile Information Device Profile)</td><td rowspan="2">PDAP
(Personal Digital Assistant Profile)</td><td>Personal Profile</td></tr>
<tr><td>Personal Basis Profile</td></tr>
<tr><td colspan="2">CLDC
(Connected, Limited Device Configuration)</td><td>CDC
(Connected Device Configuration)</td></tr>
<tr><td colspan="3">J2ME
(Java 2, Micro Edition)</td></tr>
</table>

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: CDC (JSR 36, 218) [5]

CDC: Connected Device Configuration

For set-top boxes, car navigation systems, smart communicators, high end PDAs, etc.

- 32bit microprocessor
- 2 MB RAM and 2.5 ROM for Java Runtime Environment

Full Java Virtual Machine

- CDC 1.0a based on J2SE 1.3.1
- CDC 1.1 based on J2SE 1.4.2

Supports three profiles: Foundation Profile, Personal Basis Profile, Personal Profile

Outdated Personal Java is still used

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: CLDC [JSR 30, 139]

Connected, Limited Device Configuration

For small devices (e.g. mobile phone, pager, PDA) with small screen size, limited memory, slow network connection

For devices with 160 to 512KB (according to the specification) of memory for Java Platform

JVM: KVM ("Kilobyte Virtual Machine")

- Not a full standard bytecode verifier
- Adding native methods not allowed → not possible to access platform-specific functionality

CLDC 1.0 / CLDC 1.1. (Floating point data types)

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: MIDP (based on CLDC) [6, 7]

- Mobile Information Device Profile for mobile phones and pagers
- Device characteristics (according to the specification):
  - → 128KB of non-volatile memory
  - → 32KB of volatile memory (runtime heap)
  - 8KB for persistent data
  - Screen: → 94*54 pixel
  - Input capacity, Network connection
- Advantages:
  - WORA (Write Once, Run Anywhere)
  - Security (Sandbox KVM)

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: APIs in CLDC 1.0 + MIDP 2.0

MIDP 2.0

javax.microedition.lcdui
javax.microedition.lcdui.game
javax.microedition.media
javax.microedition.media.control
javax.microedition.midlet
javax.microedition.pki
javax.microedition.rms

CLDC 1.1

java.lang
java.lang.ref
java.io
java.util
java.microedition.io

APIs are restricted when compared with J2SE

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Example Nokia 6600 (June 2003)

Stack:

- Configuration: CLDC 1.0
- Profile: MIDP 2.0
- Optional APIs
  - Nokia UI API
  - Wireless Messaging API (JSR-120)
  - Mobile Media API (JSR-135)
  - Bluetooth API (JSR-82 no OBEX)

Memory (6 MByte internal memory):

- Heap size: 3 MByte; Shared Memory for Storage: 6 MB + MMC; Max JAR Size: Memory allocated dynamically

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Example Nokia 6630 (June 2004)

Stack:

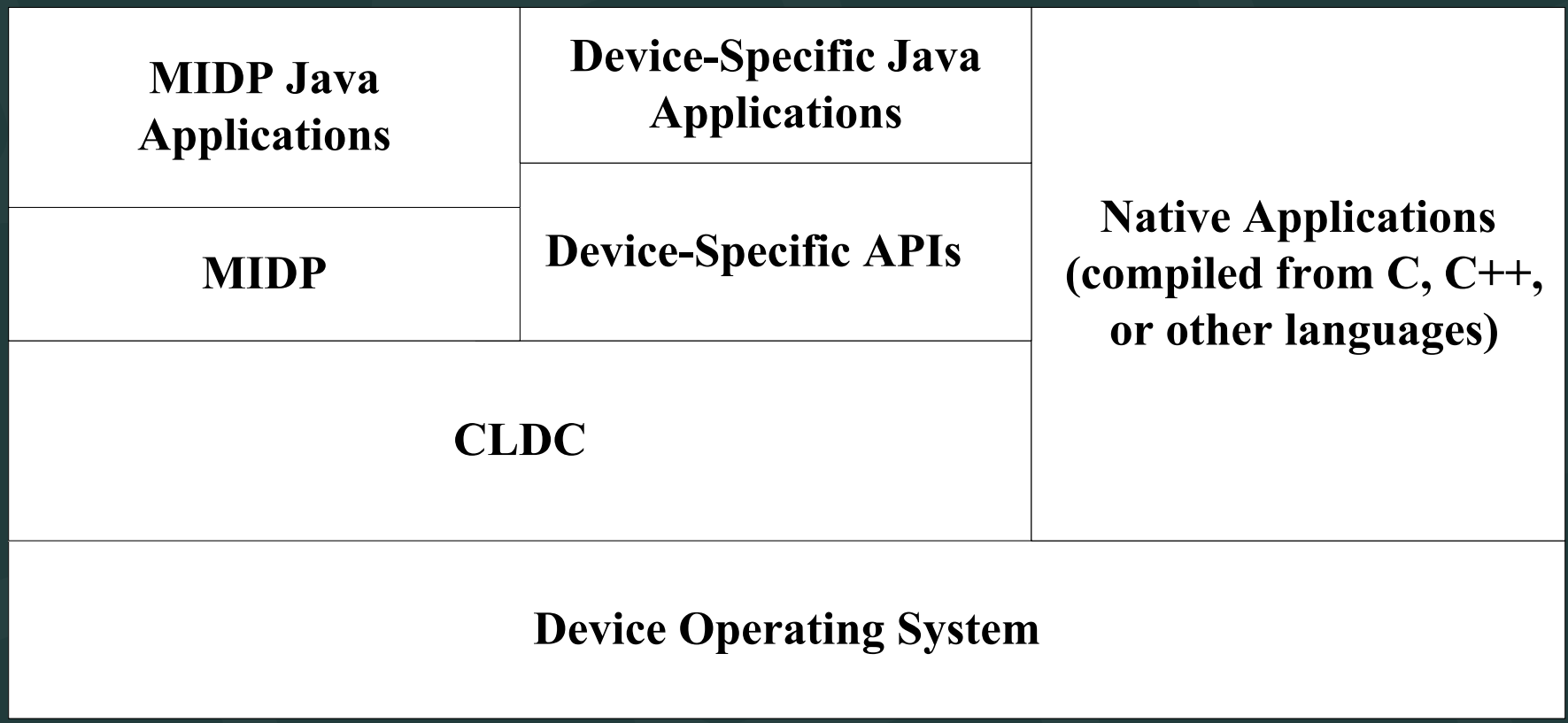
- Configuration: CLDC 1.1
- Profile: MIDP 2.0
- Optional APIs
  - APIs of 6600
  - FileConnection and PIM API (JSR-75)
  - Mobile 3D Graphics API (JSR-184)

Memory (10 MByte internal memory): :

- Heap size: Memory allocated dynamically; Shared Memory for Storage: 10 MB + MMC; Max JAR Size: Memory allocated dynamically

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

J2ME: Compatibility



Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDlet

MIDP applications are called MIDlets

Every MIDlet is instance of javax.microedition.midlet.MIDlet

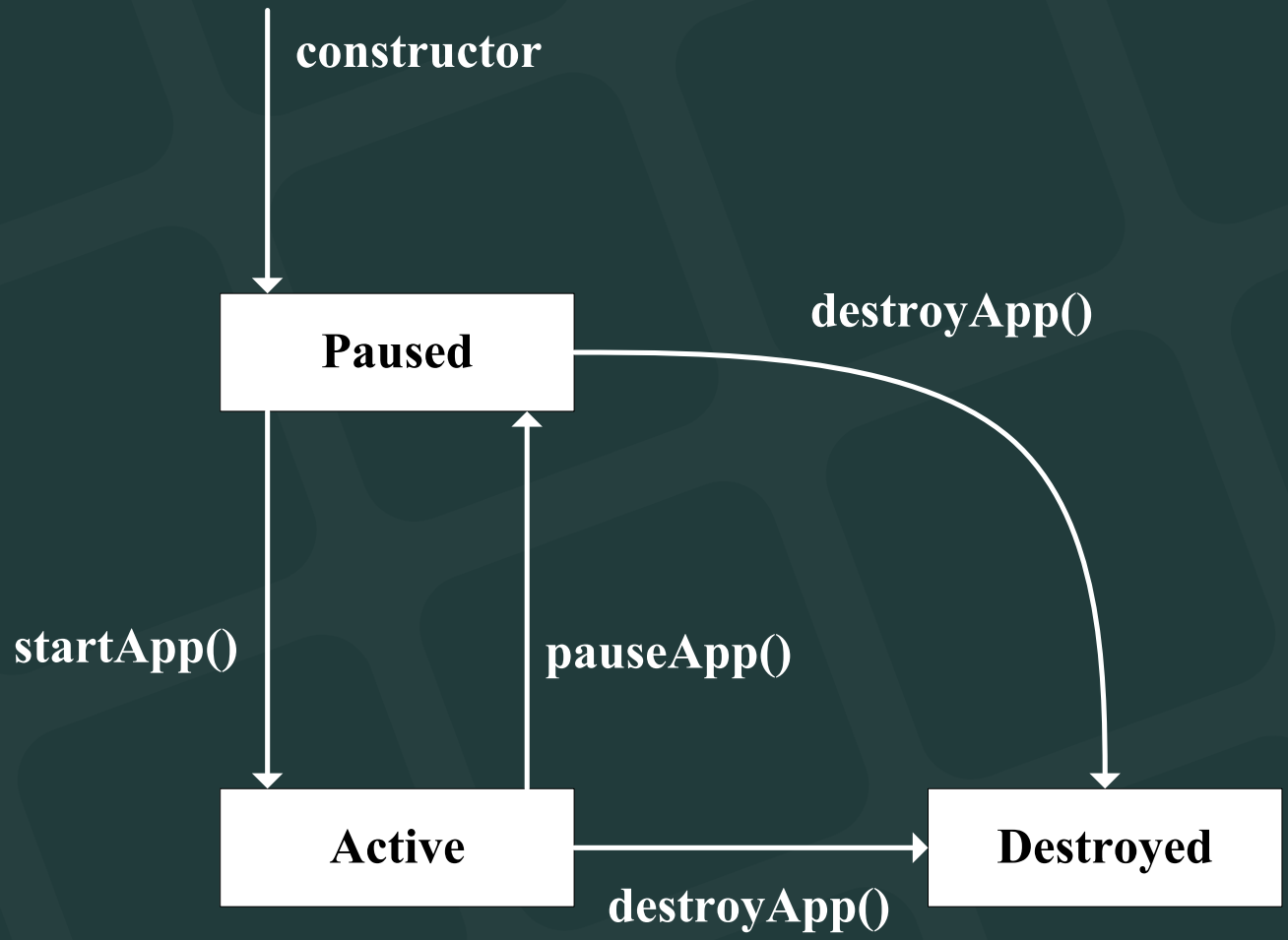
- No argument constructor
- Implements lifecycle methods

Conceptually similar to Applets

- Can be downloaded
- Executed in host environment

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDlet (MIDP Application): Life Cycle



Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDlet (MIDP Application): Life Cycle

- Application Manager: controls the installation and execution of MIDlets
- Start of a MIDlet: constructor + startApp (done by Application Manager)
- MIDlet
  - place itself in Paused state (notifyPaused())
  - destroy itself (notifyDestroyed())

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDlet (MIDP Application): Life Cycle

Application Manager

- pauseApp() and destroyApp() could be triggered by Application Manager

‘ active’ Paused state

- resumeRequest() – MIDlet wants to become Active

Methods for every state transition

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDlet Build Cycle (1/2)

(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)
- Security problem
- Normally not visible for the programmer

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDlet Build Cycle (2/2)

(4) (Application) Package, MIDlet Suite

- MIDlets + Classes + Ressources + 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
  - used for installation

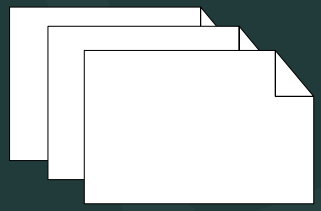
(5) Test or Deploy

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

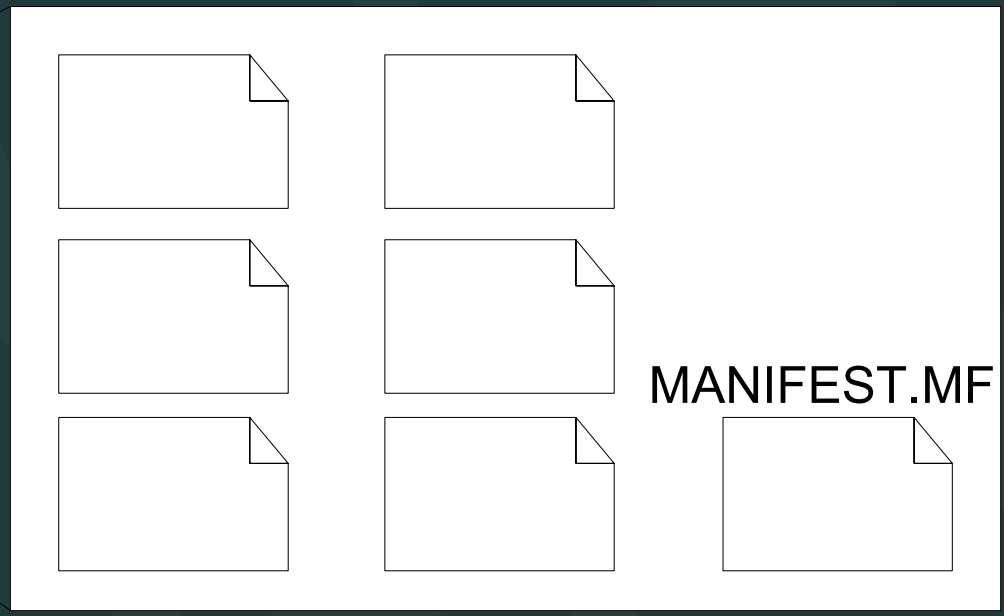
Anatomy of a MIDlet suite

MidletSuite.jad

MidletSuite.jar



Contents of MidletSuite.jar



Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

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.)

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

User Interface: Methodology

Abstraction (→ Preferred Method)

- specifying a user interface abstract terms
- (Not:) "Display the word ' Next' on the screen above the soft button."
- Rather: "Give me a Next command somewhere in this interface"

Discovery (→ Games)

- Application learns about the device + tailors the user interface programmatically
- Screen size → Scaling

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

User Interface: View from the Top

User-interface classes *javax.microedition.lcdui*

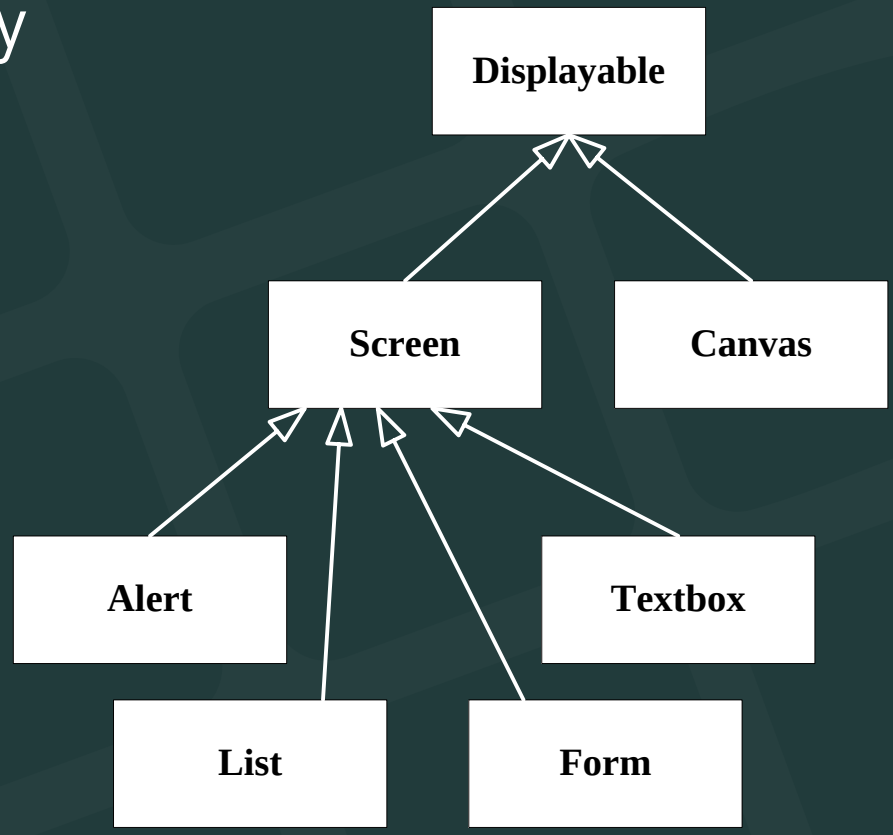
Device display represent by *Display (getDisplay())*

Display: easel

Displayable: canvas on easel

Canvas: Discovery

Screen: Abstraction



Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

User Interface: View from the Top

Changes the contents of the display: passing *Displayable* instances to *Display's setCurrent()*

Typical Sequence

- Show a *Displayable*
- Wait for input
- Decide what *Displayable* should next
- Repeat

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

User Interface: Simple Example

```
public class Commander extends MIDlet {
  public void startApp() {
    Displayable d = new TextBox("TextBox", "Commander", 20, TextField.ANY);
    Command c = new Command("Exit", Command.EXIT, 0);
    d.addCommand(c);
    d.setCommandListener(new CommandListener() {
      public void commandAction(Command c, Displayable s) {
        notifyDestroyed();
      }
    });

    Display.getDisplay(this).setCurrent(d);
  }

  public void pauseApp() {}

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

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

MIDP: Persistent Storage [8]

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’)

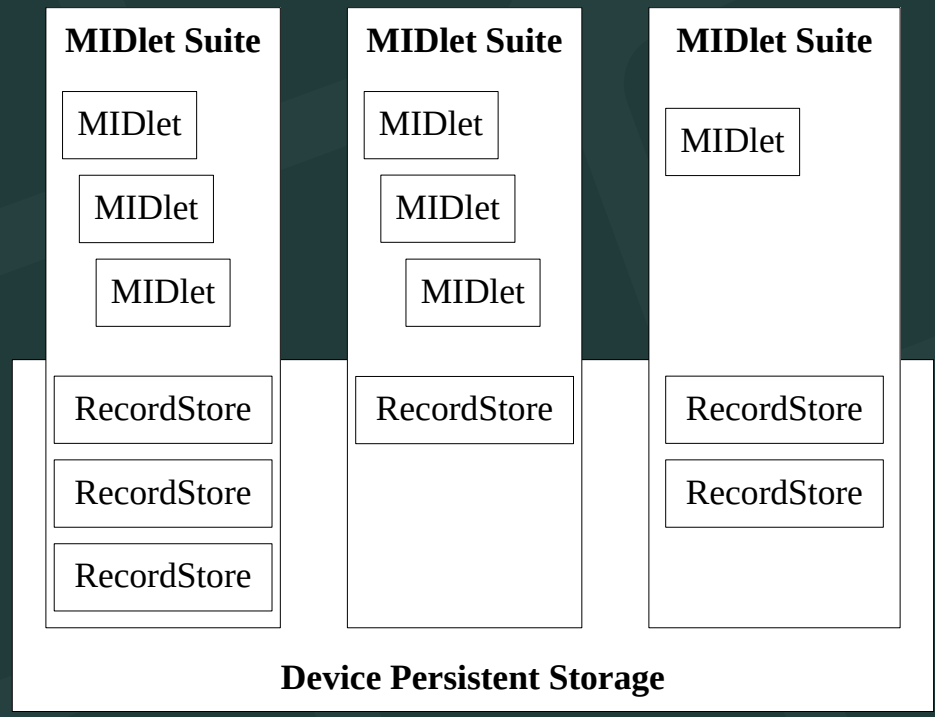
Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Persistent Storage: Records

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 !!!



Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Further APIs

Wireless Messaging API (JSR-120)
Mobile Media API (JSR-135)
Bluetooth API (JSR-82 no OBEX)
FileConnection and PIM API (JSR-75)
Mobile 3D Graphics API (JSR-184)
Location API (JSR-179)

Further APIs (not JSRs): kXML, Parsing of GPS data, etc.

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Tool Support / Development Kits

Sun's MIDP reference Implementation
Sun J2ME Wireless Toolkit (WTK 2.2)
IDE

- Netbeans
- Eclipse
- Borland JBuilder MobileSet
- IBM WebSphere Studio Device Developer
- Metrowerks Code Warrior Wireless Studio
- Sun ONE Studio, Mobile Edition

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Programming "Hello World"

IDE: Netbeans

JavaDoc

Installation on mobile phone (video)

HCILab

http://www.hcilab.org/resources/tutorials.htm

Form.nokia.com

http://www.forum.nokia.com/main.html

Siemens Mobile

http://www.siemens-mobile.com/developer

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Experiences

- CLDC/MIDP is a powerful platform for building novel application on mobile devices
- Everything (phones, APIs, tools, books, documentation, etc) is getting better in a very fast way
- Programming with J2ME It is still a novelty for most people.
- New APIs (Mobile Media, Bluetooth, etc.) have new bugs. “Old” APIs (storage, UI) are already in a matured state.

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Experiences

- Different mobile devices have different KVMs (with different bugs)
- Testing of applications on the mobile phone (!!!) is very important.
- Big differences between the emulators and the real phone.
- Lack of memory and processing power is still a problem.
- Debugging on the mobile phone is a big problem. (No meaningful error messages.)

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Experiences

Implementation on an abstract level. Not so much possibilities like in Symbian.

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Wishes for the next CLDC/MIPD Generation

New security model for J2ME

- Accessing data (record stores)
- Accessing the camera, microphone, network, Bluetooth

Quality of service

- Uncertain behavior when recording (quality, encoding) and playing (Which player?) media

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Market

Mobile devices

- are an exploding market
- because of increasing processing power, available memory and internet connectivity → attractive platform

Most supported platform on mobile devices: J2ME

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

References

[1] Jonathan B. Knudsen. Wireless Java: Developing with J2ME. Second Edition. ISBN: 1590590775.

[2] Stephen Neal. Overview of J2ME and Nokia APIs. Sun Tech Days.
http://www.nokia.co.jp/forum/publish/documents/Tech_Days_Yokohama_Workshop_Session.pdf

[3] J2ME datasheet
http://java.sun.com/j2me/j2me-ds.pdf

[4] Heise Newsticker
http://www.heise.de/newsticker/

[5] CDC Data Sheet.
http://java.sun.com/j2me/docs/j2me_cdc.pdf

[6] What's in MIDP 2.0: A Guide for Java Developers
http://www.forum.nokia.com/ndsCookieBuilder?fileParamID=3632

[7] MIDP 2.0: An Introduction
http://www.forum.nokia.com/ndsCookieBuilder?fileParamID=3231

[8] Understanding the Record Management System
http://developers.sun.com/techtopics/mobility/midp/articles/databaserms/

Enrico Rukzio
Enrico.Rukzio@ifi.lmu.de

Material

Jonathan B. Knudsen. Wireless Java: Developing with J2ME. Second Edition. ISBN: 1590590775. 2003.

Jonathan B. Knudsen. Beginning J2ME. ISBN 590594797. 30.04.2005.

Java.Sun.Com (Documentation, Code samples & Articles, FAQs, white papers, technical articles, etc.)

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

Forum.nokia.com (Documents, Code & examples, tools, forum)

http://www.forum.nokia.com

Links to documentations and tutorials at hcilab.org

http://www.hcilab.org/documents/tutorials/DocuTuto/index.html

Tutorial: Using SVN / Subversion

SVN: What is version control?

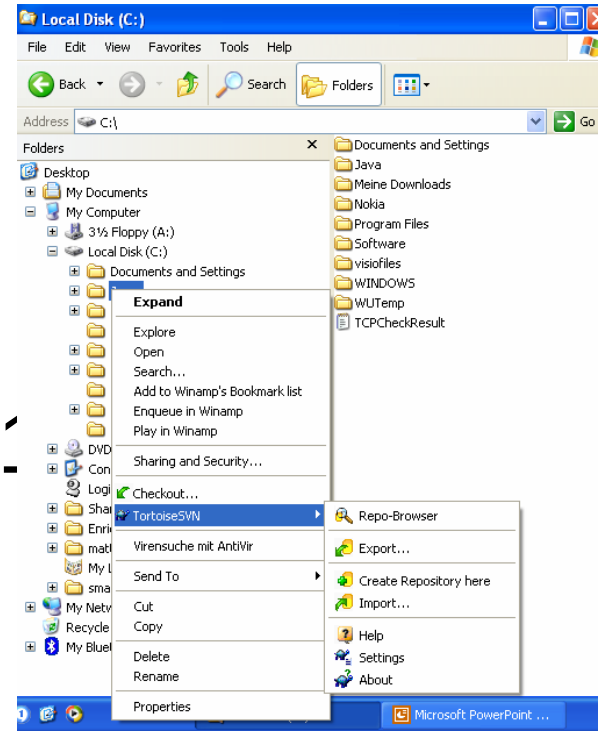
- Allows common editing of source code files (e.g. *.java) and other files
- There is one central repository, access over the network
- Work is done on a local copy, not directly on the server
- System keeps copies of all current and previous versions of files:
  - Access to old file versions + state of the project on a specific time
  - Through „Diffs” it is possible to show the difference between two versions of a (text) file

SVN: Subversion / SVN

- Successor of CVS (concurrent version system): Similar handling, Eliminates some architectural problems, better network accessing possibilities, Open Source, available for different operating systems
- Preparation/1st step: „Checkout”, d.h. getting a local copy of the current state of a repository which is transmitted from the server to the local PC
- „Update” – Update the local working copy. If for instance a other person has worked on a file and has this file already checked in the server, your local copy get through this command updated.
- „Commit” – Local changes (a file has been changed) are committed/transferred to the server

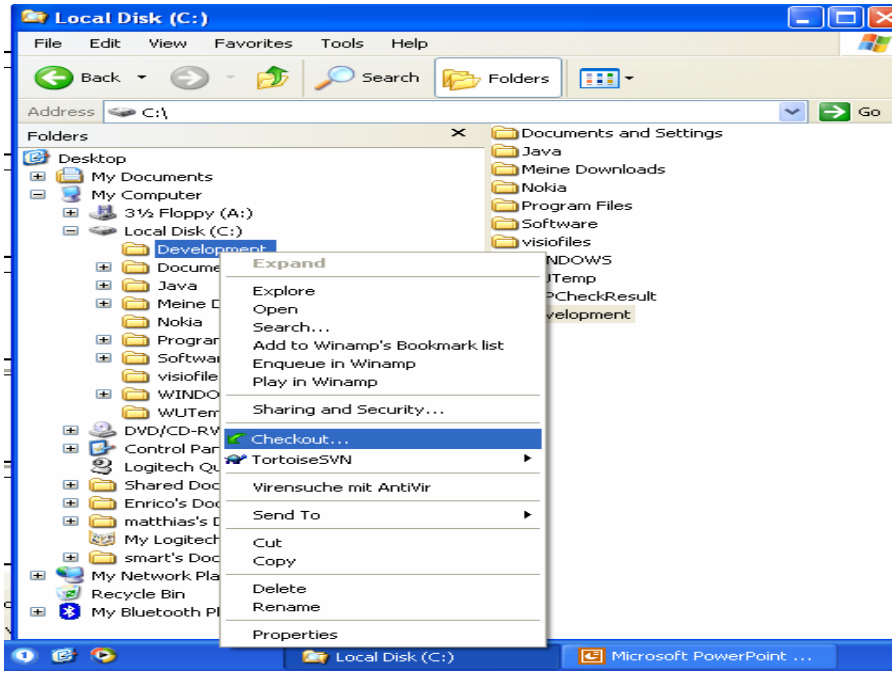
SVN: Installation of a Client

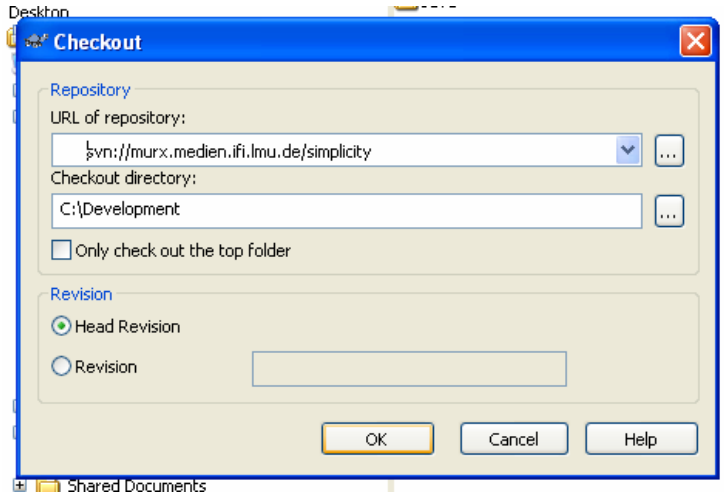
- Installation of Subversion packages
  - http://subversion.tigris.org/project_packages.html
  - http://subversion.tigris.org/servlets/ProjectDocumentList?folderID=91 (for Windows)
- Installation of TortoiseSVN
  - TortoiseSVN is a Windows client für SVN which is integrated in the explorer
  - http://tortoisesvn.tigris.org/download.html
- Already installed in 103 / 10



SVN: Checkout

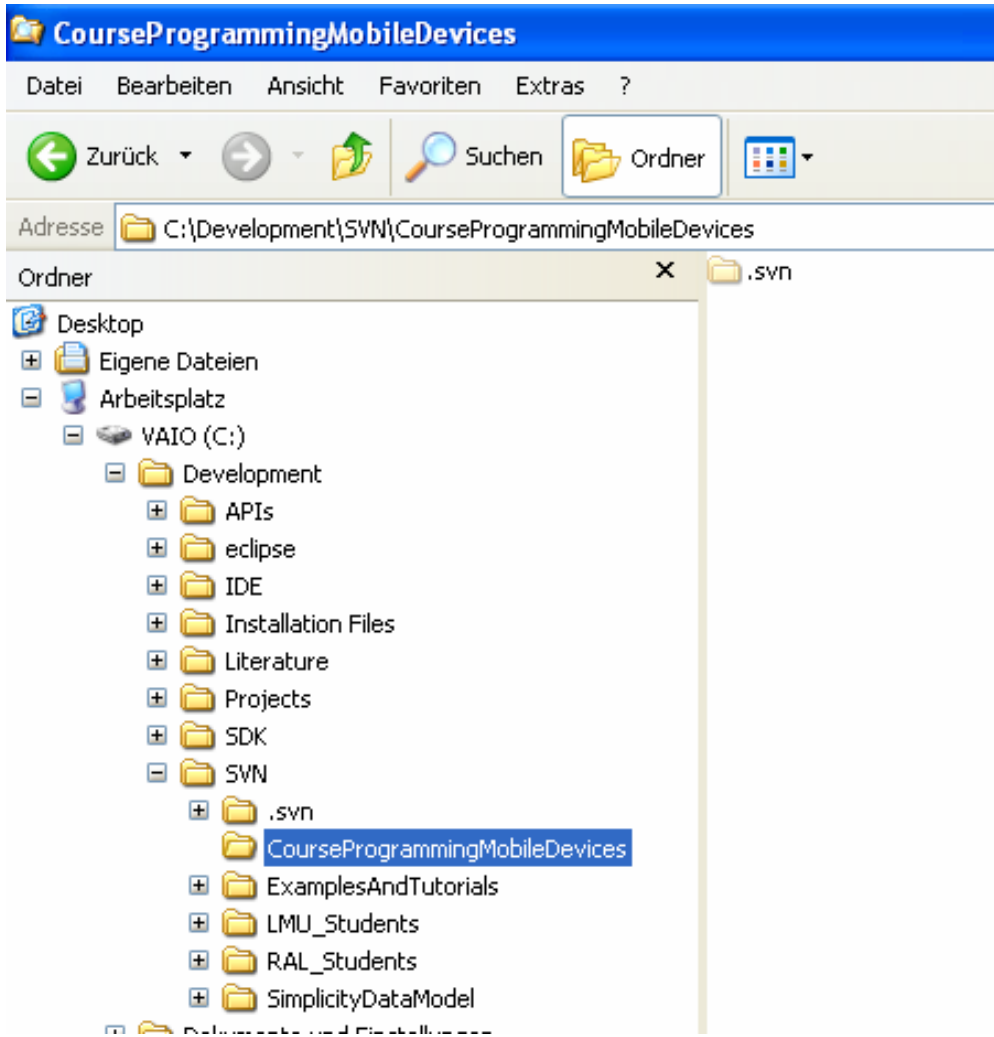
- Choose „Checkout” on a empty directory (getting a local copy of the repository)
- svn://murx.medien.ifi.lmu.de/simplicity





SVN: Checkout

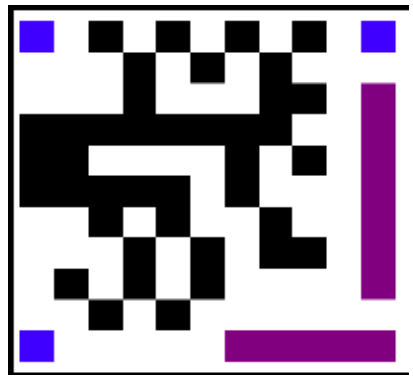
- Everybody gets a password + login
- An own directory for the practical course „PEWMS05“



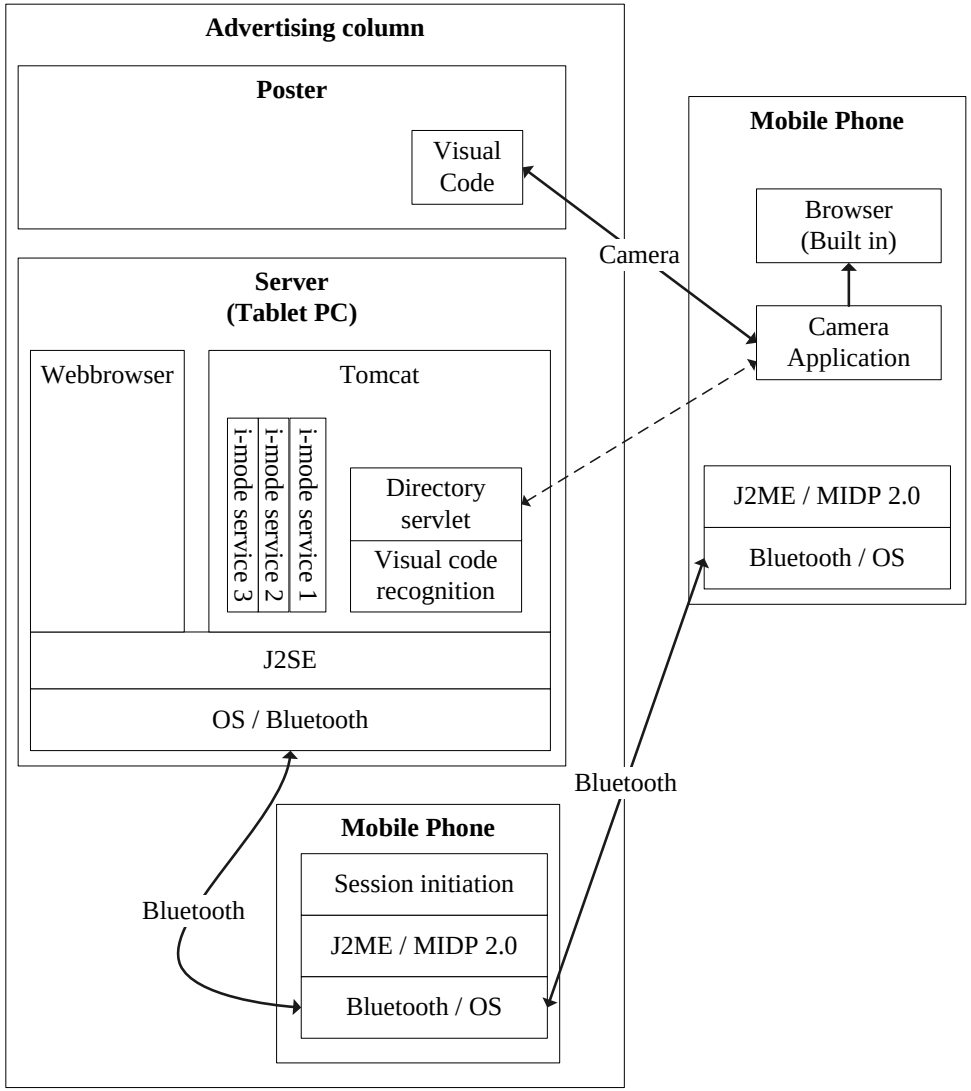
Visual Code

http://www.inf.ethz.ch/personal/rohs/visualcodes/

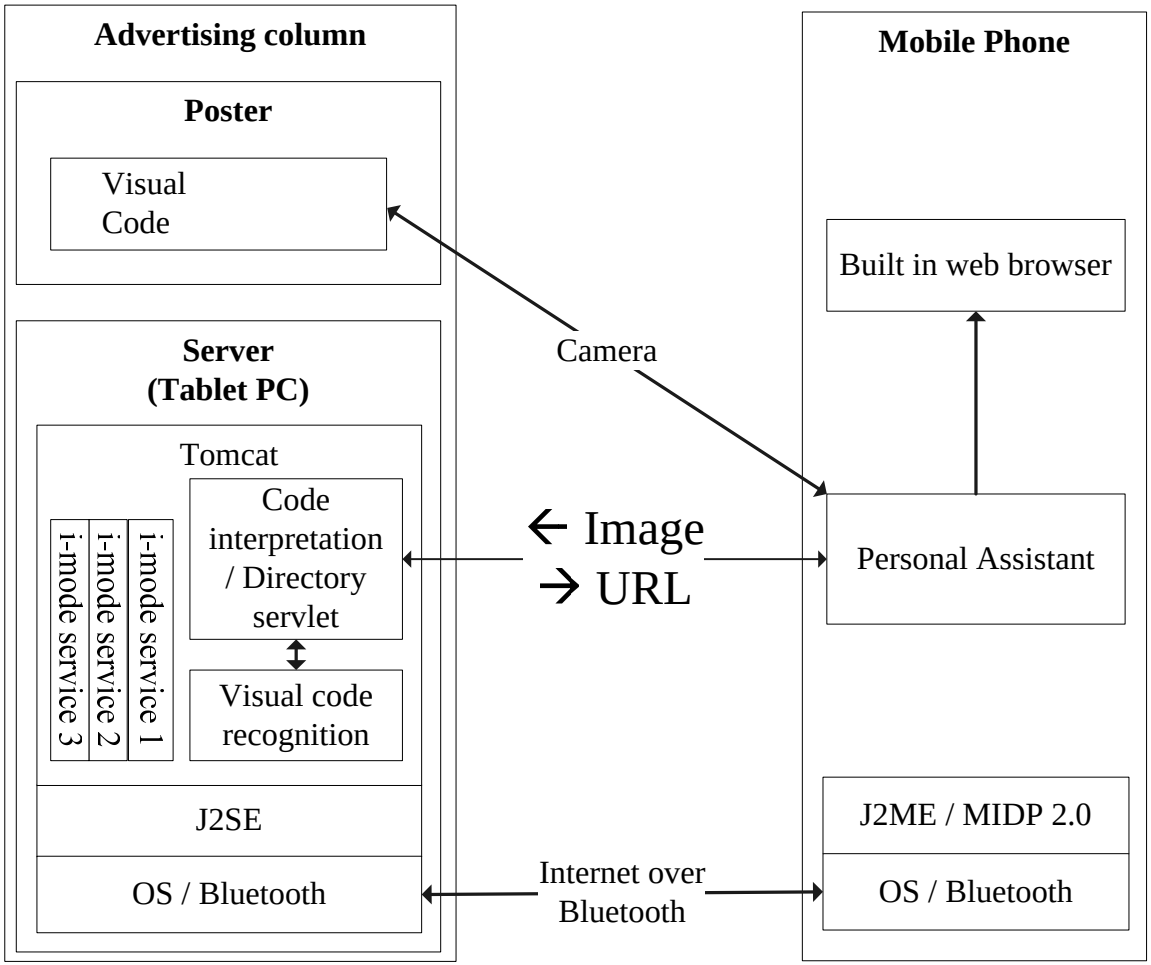
- Represents a decimal or hexadecimal number
- Used API is searching for the two guide bars and the three cornerstones
- 83 Code bit elements
- Idea: Mapping a decimal number to an URL



Group 1: Architecture



Physical Posters as Gateways: Architecture & Pr



Server class `PostServelt`

- Receives the image data as an inputstream from the mobile phone
- Creates a JPEG file

```
while ((line = r.readLine())!=null) {
        buf.append(line);
 }
 String string = buf.toString();
 byte[] data = Base64.decode(string);
  ....

 FileOutputStream file = new FileOutputStream("C:/Image/image.jpeg");
 file.write(data);
```

Server class `PostServelt`

- Calls the class `CompactRecognizerUI` from package `recognizer`
- Returns the mapped URL to the mobile phone

```
String url = CompactRecognizerUI.doDetection("C:/Image/image.jpeg");
 OutputStream out = response.getOutputStream();
 out.write(url.getBytes());
```

Server class `CompactRecognizerUI`

- Maps the visual code number to an URL
- E.g. The decimal `100` should map the URL **`http://www.mimuc.de`**

```
public static void initURL(){
        urlVector.add(new URLCode(100, "http://www.mimuc.de"));
        ......
}
```

Client class `MainController`

- Midlet class which controls the whole application
- ALSO INCLUDES CODE FOR THE SERVICE ACCEPTOR!!
- Creates class `MarkerDetector`
- Displays the web page/video

```
public void browserRequest(String url) throws ConnectionNotFoundException{
        if (platformRequest(url)) {
            destroyApp(false);
        }
    }
```

Client class `MarkerDetector`

- Creates `MarkerDetectorForm` and `MarkerDetectorErrorForm`
- Transmits the taken picture to the server

```
String buffer = Base64.encode(byteArray of theImage);
String servletUrl = new String(„http://ip-address:port/path/PostServlet");
HttpConnection hc = (HttpConnection)Connector.open(servletUrl);
hc.setRequestMethod(HttpConnection.POST);
OuptputStream out = hc.openOutputStream();
out.write(buffer.getBytes());
```

Client class `MarkerDetector`

- The mobile phone receives the mapped URL

```
in = hc.openInputStream();
while ((ch = in.read()) != -1) {
      responseBuf.append((char)ch);
 }
m_mainController.browserRequest(responseBuf.toString());
```

Client class `MarkerDetectorForm`

- Contains all code needed for taking a picture
- Takes control of the built-in camera

```
Player player=Manager.createPlayer(“capture://video”);
player.realize();
...
VideoControl  videoControl=(VideoControl)player.getControl(“VideoControl”);
 ...
append((Item)(videoControl.initDisplayMode(VideoControl.USE_GUI_PRIMITI
    VE,null)));
...
player.start();
```

Client class `MarkerDetectorForm`

- Taking a picture

```
byte[] pngImage =
    videoControl.getSnapshot("encoding=jpeg&width=200&height=160");
```

- Displays a progress bar during the picture transmission

Client class

`MarkerDetectorErrorForm`

- Contains all code for an error handling
- Displays the taken picture and shows the occured error

Conclusion

- Server Classes
  - `PostServlet`
  - `Base64`
  - **Packages:** **`recognizer`** and **`hammingcode`**
- Client Classes
  - `MainController`
  - `MarkerDetector`
  - `MarkerDetectorForm`
  - `MarkerDetectorErrorForm`
  - `Base64`

Resources

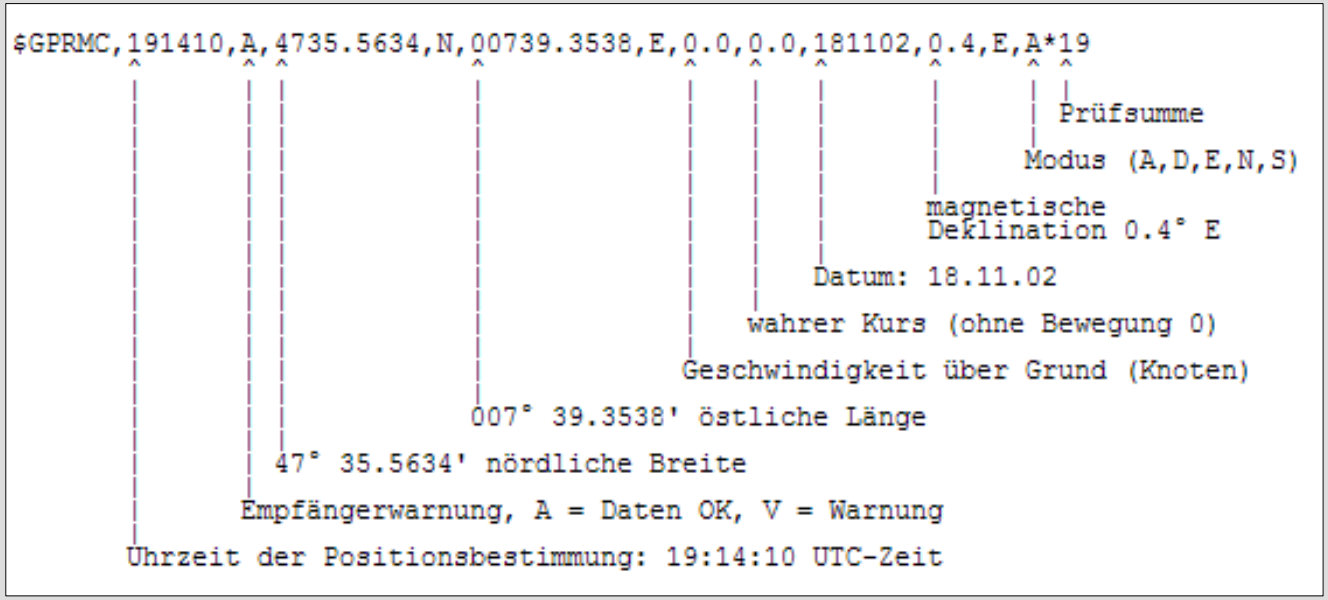
- **Download `PME_GROUP1_WS0405.zip`**
- **It contains the code, a documentation and other stuff.**
- **The zip file is available at the practical course directory GROUP1 for PME in the winter term 2004/05.**

GPS Bluetooth Empfänger
Hardware

- Kommunikation via Bluetooth mit dem Handy
- Externe Antenne für besseren Empfang (Amalienstraße: 5 – 7 Satelliten)

GPS Bluetooth Empfänger
Datenformat

- Protokoll-Standard: NMEA-0183 (National Marine Electronics Association)
- Übertragung im ASCII-Format zeilenweise in Datensätzen (GPRMC, GPGGA, ...)



GPS Bluetooth Empfänger
Kommunikation

- Geräte koppeln
- Verbindung aufbauen

```
String url = "btspp://0002c727fc56:1";

javax.microedition.io.StreamConnection connection = (StreamConnection) Connector
        .open(url, Connector.READ);

java.io.InputStreamReader reader = new InputStreamReader(connection
        .openInputStream());
```

(Bluetooth-Adresse ist bekannt)

- Daten einlesen

```
String output = "";
int input;

while ((input = reader.read()) != 13)
    output += (char) input;
```

GPS Bluetooth Empfänger
Einbindung I

- Klasse **GPS**
  - Läuft im Hintergrund als Thread (mit **start()** aktivieren, mit **stop()** untebrechen)
  - Baut Verbindung zum GPS-Empfänger auf, liest eine Serie von Datensätzen ein und parsed diese
  - Baut die Verbindung wieder ab (Puffer-Problem), wartet einige Sekunden und baut die Verbindung im Anschluss wieder automatisch auf
  - Über **getRecord()** werden die zuletzt eingelesenen Daten abgerufen

GPS Bluetooth Empfänger
Einbindung II

- Klasse **Parser**

```
/**
 * Parses a string sent by GPS receiver.
 *
 * @param s
 *            String to be parsed
 * @param record
 *            Record to store data
 * @return Type of record
 * @throws UnsupportedTypeException
 *             If type is not recognized
 * @throws ParseException
 *             If there was an error during parsing
 */
public static int parse(String s, Record record)
        throws UnsupportedTypeException, ParseException {
```

- Klasse **Record**
  - Speichert die geparseten Daten

```
public String courseMadeGood = "";
public String dateTimeOfFix = "";
public String groundSpeed = "";
public String lattitude = "";
public String lattitudeDirection = "";
public String longitude = "";
public String longitudeDirection = "";
public String magneticVariation = "";
public String quality = "";
public String satelliteCount = "";
public boolean warning;
```