Structure

- Basic conditions of the practical course
- Group 1: Advertising column (8 Students)
- Group 2: Photo treasure hunt (5 Students)
- Group 3: Support for traffic wardens (5 Students)
- Organizational aspects
  - Who is in which group
  - Next meeting
- Tutorial: Usage of Tortoise / Subversion
- Tutorial: Developing J2ME applications
Basic Conditions

- Software development as teamwork
- Management and self-organization
  - Project and time management
  - Plan → result
  - Specification and documentation
- Project schedule / Requirement specification
  - Who does what and when?
  - Work packages, milestones, dependencies, structuring the tasks

Basic Conditions

- Predetermined
  - result
  - generic work packages
  - technical scope
  - there should be a kind of organization
- Official meetings every 2 weeks
  - 21/10, 04/11, 18/11, 02/12 (?), 16/12 (preliminary report), 13/01, 27/01, 10/02 (final report)
Group 1: Posters as Gateways

- Idea: Physical Posters as Gateways to Context-aware Services for Mobile Devices [1,2]
- Usage of built-in camera of the mobile phone, Marker on poster represents service (URL)
- Bridge between the real and the virtual world
- Network: Could be provided by the advertising company (integrated Bluetooth-access point in an advertising column)

Group 1: Posters as Gateways

- Scenario
  - Posters on advertising columns
  - “Killing time is a killer application”

- User study
  - Observed 230 passengers at stops
  - How long do they wait?
    - Interval between 2 Busses = 5 Minutes
    - 44% were waiting more then 3 min
  - What do they do?
    - nothing, be bored, talking (max 8-20%), calling, reading
Group 1: Work packages

- WP 1: Architecture
  - Define basic architecture
  - Define communication mechanisms
- WP 2: Basics
  - Mobile, lightweight advertising column
  - Build 3 mobile services (HTML, i-mode, WAP)
    - Limited offers (book a flight (e.g. Gexx))
    - Location based (reserve a hotel room)
    - Technical product information (Mediamarkt)
    - Optional: one should provide a video
  - Attach posters, Having posters
  - Test the interaction
  - Performance & Robustness
- WP 3: Integration of a display
  - Tablet PC as a part of the advertising column
  - Enhanced poster: Mobile phone as a remote control
  - Same content on the tablet pc and the mobile phone
- WP 4: Session initiation
  - Who makes the first contact? (person or poster)
  - Interaction is initiated by the poster / advertising column
- WP 5: Adaptation
  - User Preferences
    - session initiation
    - (semi) automatic selection of the network (Bluetooth or GPRS)
  - Based on the location of the advertising column (hotel room in this city, flight from this city, next mediamarkt)
  - Context (e.g. Daytime, date, weather)
- WP 6: User study
  - User study
  - Compare: type in URL, short number, get URL from marker, proactive posters, enhanced posters (display)
Group 1: Architecture: WP 1

**WP1: Basics**

- **Advertising column**
  - Visual Code
  - Camera

- **Server (Tablet PC)**
  - Tomcat
  - Directory servlet
  - Visual code recognition
  - J2SE
  - OS / Bluetooth

- **Mobile Phone**
  - Browser (Built in)
  - Visual Code Recognition Midlet
  - J2ME / MIDP 2.0
  - Bluetooth / OS

Group 1: Architecture: WP 2

**WP2: Integration of a Display**

- **Advertising column**
  - Visual Code
  - Camera

- **Server (Tablet PC)**
  - Webbrowser
  - Tomcat
  - Directory servlet
  - Visual code recognition
  - J2SE
  - OS / Bluetooth

- **Mobile Phone**
  - Browser (Built in)
  - Camera Application
  - J2ME / MIDP 2.0
  - Bluetooth / OS
Group 1: Architecture: WP 3

WP2: Session initiation

Group 2: Photo Treasure Hunt

• Basic idea:
  - players should look for and take/collect a defined set of images
  - The images are to be uploaded to a web site
  - application that can be customized by a task description

• Steps
  - Editing a hunt on a website
    - Describing the pictures that should be taken
    - Providing this description via XML/HTTP
    - Set a start time and duration for the game
  - Players download the game and run it
    - Get a list of photos to take
    - Take photos and associate them with the list
    - (trade photos with other players, mobile Bluetooth)
  - Goal
    - Have a complete set of photos to match the list by the end of the game
    - Upload it to the web site before the end of the game
    - have it on display and get voted the winner

• Evaluate the game based on the following scenarios
Group 2: scenarios

- **Town game**
  - the game authors creates a list of interesting buildings (e.g. the towers of Frauenkirche, the entrance to the Olympic Stadium, someone riding the waves in the Englische Garten, …)
  - Visitors at the town information centre download the game and play it
  - The uploaded pictures are on the web – and other users can vote one for winner

- **Party Game**
  - The game author creates a list of interesting pictures (e.g. a red size 38 shoe, someone smoking lucky strike, …)
  - …

Group 2: Photo Treasure Hunt

- **WP 1: Architecture**
  - Define basic architecture
  - Define communication mechanisms

- **WP 2: Game editor Web page**
  - Design and build web page
  - Test game design

- **WP 3: Mobile Phone Application**
  - Download task list
  - Take pictures and associated them with the task list
  - Store pictures for upload and trading
  - View pictures

- **WP 4: Backend Web site**
  - Upload of annotated photos
  - Display of uploaded images
  - Voting function

- **WP 5: Sharing of Photos over Bluetooth**
  - Bluetooth connection
  - Exchange / negotiation, 1-for-1 or 1-for-n

- **WP 6: User study**
  - Test the system in single player mode
  - Test the system with sharing
Group 3: Traffic warden support

- **Basic idea**
  - Support for traffic wardens (Kommunale Polizeihilfskraft)
  - Tasks of a traffic warden
    - Find cars that are not correctly parked
    - Filling out a parking ticket (time, place, car number, describing the situation, etc.)
  - Support through a mobile phone
    - Taking pictures of the location and the car (automatically: time, place (GPS), car number, pictures)
    - Transmission of data to a web server

- **WP 1: Architecture**
  - Getting the requirements
  - Define basic architecture
  - Define communication mechanisms

- **WP 2: GPS Coordinates on the mobile phone**
  - Establish a Bluetooth connection
  - Parsing of the transmitted information

- **WP 3: Optical Character Recognition**
  - Transfer of the image to a server (if needed)
  - Optical Character Recognition
  - Feedback if car number has been detected by OCR

- **WP 4: Backend Web site**
  - Upload data (photos, GPS, time, further information, etc.)
  - Display of parking tickets

- **WP 6: User study**
  - Test the system
Organizational stuff

- 4 SWS
- Room for the practical course
  - 208, Amalienstraße 17
  - open during normal working times (7.30-17.00)
  - 1 key for every group
- Mailing lists
  - pme@medien.ifi.lmu.de
  - {pme.g1, pme.g2, pme.g3}@medien.ifi.lmu.de
- News (Reservations, Meetings, etc.)
  - http://www.medien.informatik.uni-muenchen.de/lehre/ws0405/pme.html

Room 208 / 5 PCs

- Files: C:\Devel
- Java(TM) 2 SDK, Standard Edition 1.4.2_04
- Netbeans IDE 4.0 Beta + Mobility Pack
- J2ME Wireless Toolkit 2.2
- Nokia Developer's Suite 2.2 for J2ME
  - C:\Devel\Nokia\Devices\Series_60_MIDP_Concept_SDK_Beta_0_3_1_Nokia_edition\docs\index.html
    → JavaDoc for Nokia APIs
- Subversion / Tortoise
- Working directory: \igor\pme_ws04
  - special directory for every group + one generic
Organizational stuff

- Hardware
  - Mobile Phones: Nokia 6600 (3x), Siemens S65 (2x)
  - Tablet PC (2x)
  - Bluetooth sticks (5x)

- Book: “Wireless Java: Developing with J2ME”, in Room 208

---

Organizational stuff

- Enrico Rukzio
  - Enrico.Rukzio@ifi.lmu.de
  - Room 206, Amalienstraße 17

- Albrecht Schmidt
  - Albrecht.Schmidt@ifi.lmu.de
  - Room 502, Amalienstraße 17
Next meeting

- Date 04/11/04, 12.00, Room 107
- Every group presents
  - project plan, detailed work packages
  - who does what and when
  - first results, WP 1: Architecture
- Every group
  - Testing of APIs, HelloWorld, Tutorials
  - Own running application on mobile phone
  - Familiar with developing environment / SVN

Who is in which group?

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karin Leichtenstern</td>
<td>Alexander De Luca</td>
<td>Tobias Lang</td>
</tr>
<tr>
<td>Sven Siorpaes</td>
<td>Claudia Ruch</td>
<td>Julius Bahr</td>
</tr>
<tr>
<td>Helge Groß</td>
<td>Monika Ticaloiu</td>
<td>Kai Schreiber</td>
</tr>
<tr>
<td>Frederike Otto</td>
<td>Ronald Eckar</td>
<td>Christoph Empl</td>
</tr>
<tr>
<td>Wu Wen Wang</td>
<td>Eva Vodvarsky</td>
<td>Tobias Lang</td>
</tr>
<tr>
<td>Christoph Pahre</td>
<td>Michael Müller</td>
<td>Dominik Schmidt</td>
</tr>
</tbody>
</table>

Belegungszeiten Raum 208
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 another 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
- Installation of TortoiseSVN
  - TortoiseSVN is a Windows client for SVN which is integrated in the explorer
  - http://tortoisesvn.tigris.org/download.html
- Already installed in 208
SVN: Checkout

- Choose „Checkout“ on an 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 „CourseProgramming MobileDevices“
Tutorial: Developing with J2ME

Developing of Applications for mobile Devices

- Devices: Basic Phone, Extended Phone, Smartphone, PDA, Notebook
- Operating Systems (Mobile Phone, Smartphone)
  - Platform specific: Symbian OS (C++, OPL), Palm OS (C++), Pocket PC, Vendor-specific
  - Platform independent: J2ME (Java 2 Platform, Micro Edition)
    - Supported by Motorola, Nokia, Panasonic, Samsung, Sharp, Siemens, Sony Ericsson, Toshiba, etc.
J2ME: Basics

- **J2ME**: Java 2 Platform, Micro Edition
  - “Java for small devices”

- **Stack**
  - Configuration + profile + optional APIs

- **Configuration**: specific kind of device
  - Specifies a Java Virtual Machine (JVM)
  - Subset of J2SE (Standard Edition)
  - Additional APIs
## J2ME: Basics

- **Profile:** more specific than configuration
  - based on a configuration
  - adds APIs for user interface, persistent storage, etc.
- **Optional APIs:** additional functionality

### Stack for Nokia 6600:
- **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)

### Stack for Siemens S65
- **Configuration:** CLDC 1.1
- **Profile:** MIDP 2.0
- **Optional APIs:** JSR 120 WMA 1.0, JSR 135 MMA 1.0, JSR 185 JTWI 1.0, JSR 184 3D API, JSR 179 Location API, JSR 82 BT API

## The J2ME universe

<table>
<thead>
<tr>
<th>Smaller</th>
<th>Larger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pagers</td>
<td>Mobile Phones</td>
</tr>
<tr>
<td>MIDP (Mobile Information Device Profile)</td>
<td>PDAP (Personal Digital Assistant Profile)</td>
</tr>
</tbody>
</table>

**Profiles:**
- **Personal Profile**
- **Personal Basis Profile**
- **Foundation Profile**
J2ME: CLDC

- 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 (statement is out of date) of memory for Java Platform
- JVM: KVM (“Kilobyte Virtual Machine”)
  - Limitations (no floating point data types)

J2ME: MIDP

- Mobile Information Device Profile
- Device (such as mobile phones and pagers) characteristics:
  - > 128KB of non-volatile memory
  - > 32KB of volatile memory (runtime heap)
  - 8KB for persistent data
  - Screen: > 94*54 pixel
- Advantages: WORA (Write Once, Run Anywhere), Security (Sandbox KVM)
J2ME: APIs in CLDC 1.0 + MIDP 2.0

**CLDC 1.0**
- java.lang
- java.io
- java.util
- java.microedition.io

**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

APIs are restricted when compared with J2SE
- Link Nokia API

---

**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
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())
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

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
MIDlet Build Cycle (2/2)

(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 an external file
  - Normally used for installation

(5) Test or Deploy

Anatomy of a MIDlet suite

- MidletSuite.jad
- MidletSuite.jar
- Contents of MidletSuite.jar
- MANIFEST.MF
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.)

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
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

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
User Interface: Simple Example

```java
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) {}
}
```

MIDP: Persistent Storage

- **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: 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 !!!

**Addressing the camera**

- Mobile Media API – JSR 135

- Example
  - Forum Nokia: Camera MIDlet - A Mobile Media API Example v1.0
  - Forum Nokia: Brief Introduction to the Mobile Media API v1.0

- Architecture
Documentation and Tutorials

- [http://www.hcilab.org/docs/mobile/](http://www.hcilab.org/docs/mobile/)
  - Internet over Bluetooth
  - Image evaluation on mobile phones
  - Picture transmission over HTTP

  - Java / Web
  - J2ME (Generic / Specific)
  - Netbeans
  - Bluetooth (JSR 82)
  - Mobile Media (JSR 135)

Development of Midlets with Netbeans 4.0 Beta 2

- New Project
  - Emulator Platform

- Mobile Application: HelloWorld Midlet
  - Methods
  - Compiling
  - Running in the Emulator

- Installing the midlet on the mobile phone
  - Transfer via Bluetooth
  - Installation, De-Installation
  - Start the application
References


