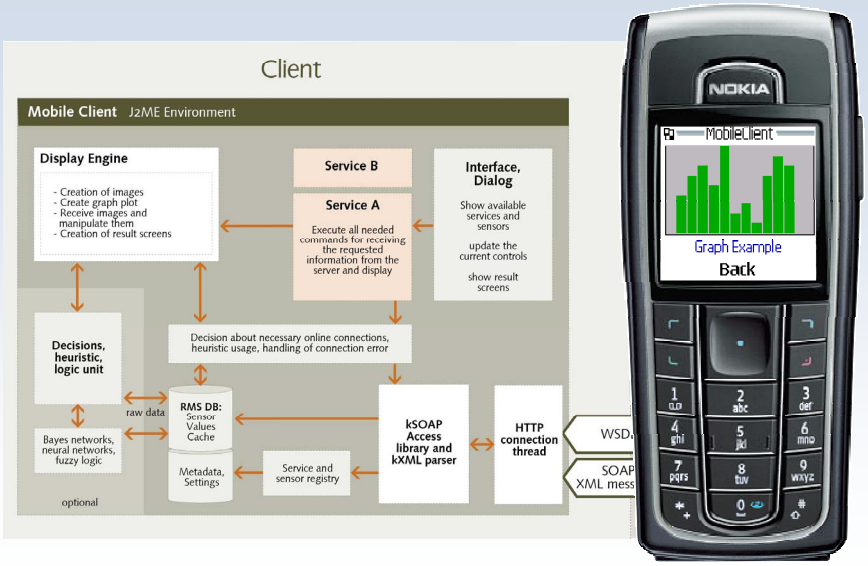
Bauhaus University Weimar



Research, Technology:
MD5, Java Sound, Bluetooth

Project Sensation

November 2004

Nicolai Marquardt
CML Cooperative Media Lab
CSCW, Prof. Tom Gross, Tareg Egla
Bauhaus University Weimar

Outline

1. Security, MD5, Architecture
2. Mobile Device 2D GraphClass
3. Java Sound and Audio
4. Bluetooth

1. Security, MD5, Architecture

1. Security, MD5, Architecture

Security Issues:

- Different security levels in our infrastructure
- Web Services
  - Web Server identification for different users
  - Server security options → Tomcat
  - Add XML security, profiles, encryption of data
  - Summary: http://java.sun.com/webservices/docs/1.2/tutorial/doc, Chapter 24
- Sockets
  - SSL – Socket Secure Layer
  - Drawback: need certificate instance, vendor
- XML-RPC, see Web Services
- Bluetooth
  - Message encryption
  - Security implemented in application layer

→ Not in Release V1 or V2, perhaps in the next semester

1. Security, MD5, Architecture

MD5 – Message Digest:

- Algorithm that takes a message (variable length) and generate a fixed-length hash values
- 128 bits MD5, 160 bits SHA-1
- If any bit of the original message changed, the hash values is completely different
- One-way hash function: computationally impossible to reconstruct the original message from the hash value
- java.security.MessageDigest (since J2SE 1.3)
- Using for Login (secure):
  1. Server sends random data
  2. Client concatenates password with random data
  3. Client use MD5 for hash value
  4. Server also concatenates the both strings and then compares his own MD5 value with the client one
- We use MD5 for key value in the facet hashtable (with interpretation preferences)

1. Security, MD5, Architecture

MD5 Method:

```
import java.security.MessageDigest;
```

```
public class MD5Calculation {

    public static void main(String[] args) throws Exception {
        byte[] messageBytes;
        String message = "userid";
        messageBytes = message.getBytes("UTF8");

        // Get a message digest object using the MD5 algorithm
        MessageDigest algorithm = MessageDigest.getInstance("MD5");

        // Calculate MD5 hash values
        algorithm.update(messageBytes);
        byte messageDigest[] = algorithm.digest();

        // Create HEX value
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < messageDigest.length; i++) {
            String hex = Integer.toHexString(0xff & messageDigest[i]);
            if (hex.length() == 1) hexString.append('0');
            hexString.append(hex);
        }

        System.out.println("\nMessage Digest: " + hexString.toString());
    }
}
```

1. Security, MD5, Architecture

Sensor Hashtable

Sensor ID 1

SensorType = Temperature

int getValue()
float getValue()
String getValue()
get HWMetadata()
getLocation()
getDescription()
getMetadata(Str)
getDataFormat()

Properties:

- Hardware ID
- Command
- **DataFormat**
- Max Value
- Min Value
- HWMetadata

Methods:
String getValue(SensorID)
String [ ] getValues(SensorID, from, until)
HardwareMetadata getHardwareMetadata(SensorID)

Int[ ] getSensors(Sensor.Location, String)
Int[ ] getSensors(Sensor.Owner, String)
Int[ ] getSensors(HardwareID)

Hardware Implementation Hashtable

Hardware ID 1

Object Instance ESB
getMethods(command, module ID)
HardwareMetadataXML myData

Hardware ID 2

ObjectInstance ESB RF
getValue(command, module ID)
HardwareMetadataXML myData

HardwareMetadata:

- int: Module ID
- String: Description
- String: Location
- String: Owner
- String: Comment
- float: degree of longitude
- float: degree of latitude
- float: above sea level
- date: Since
- date: Until

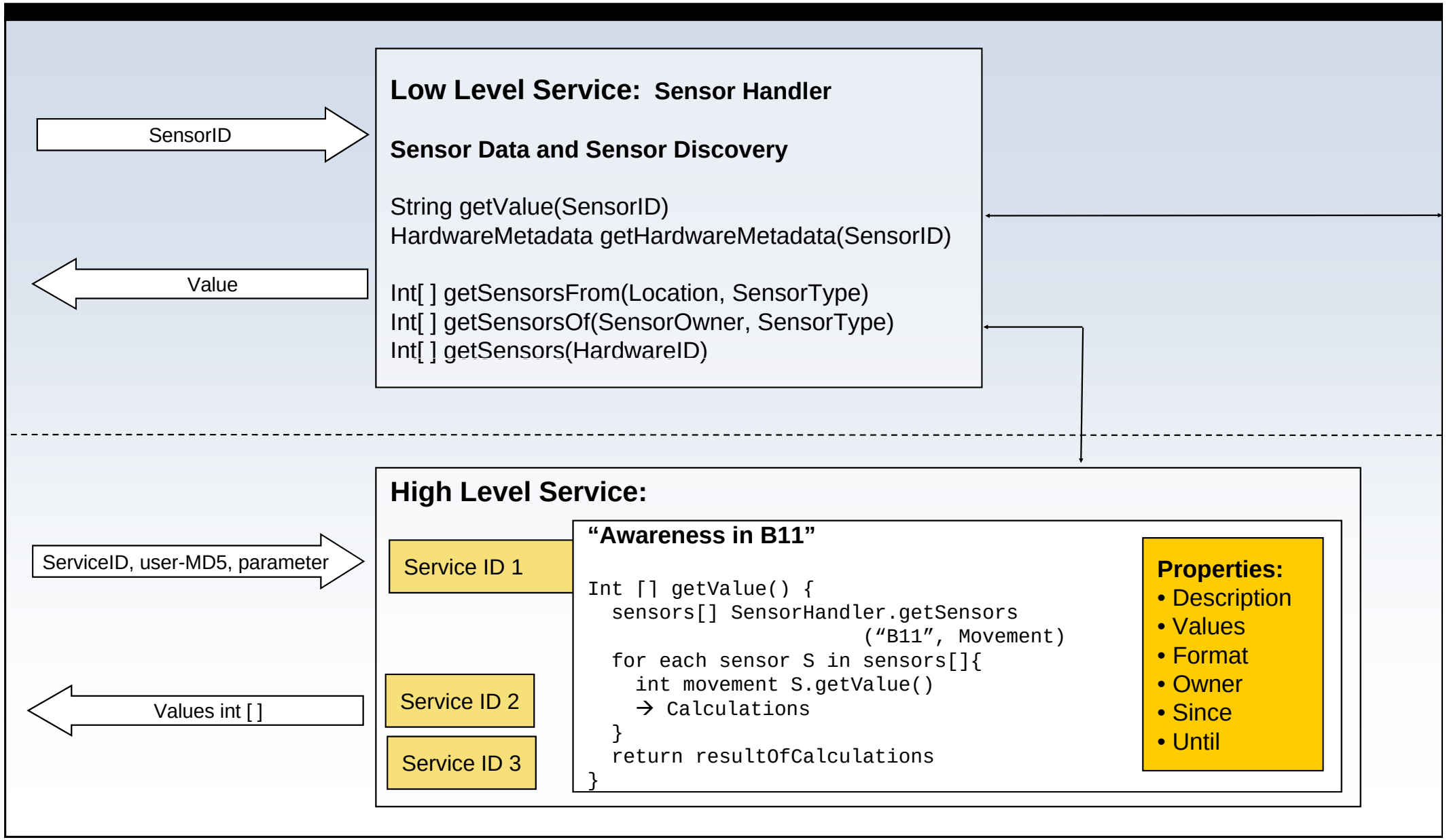
1. Security, MD5, Architecture

Commitment:

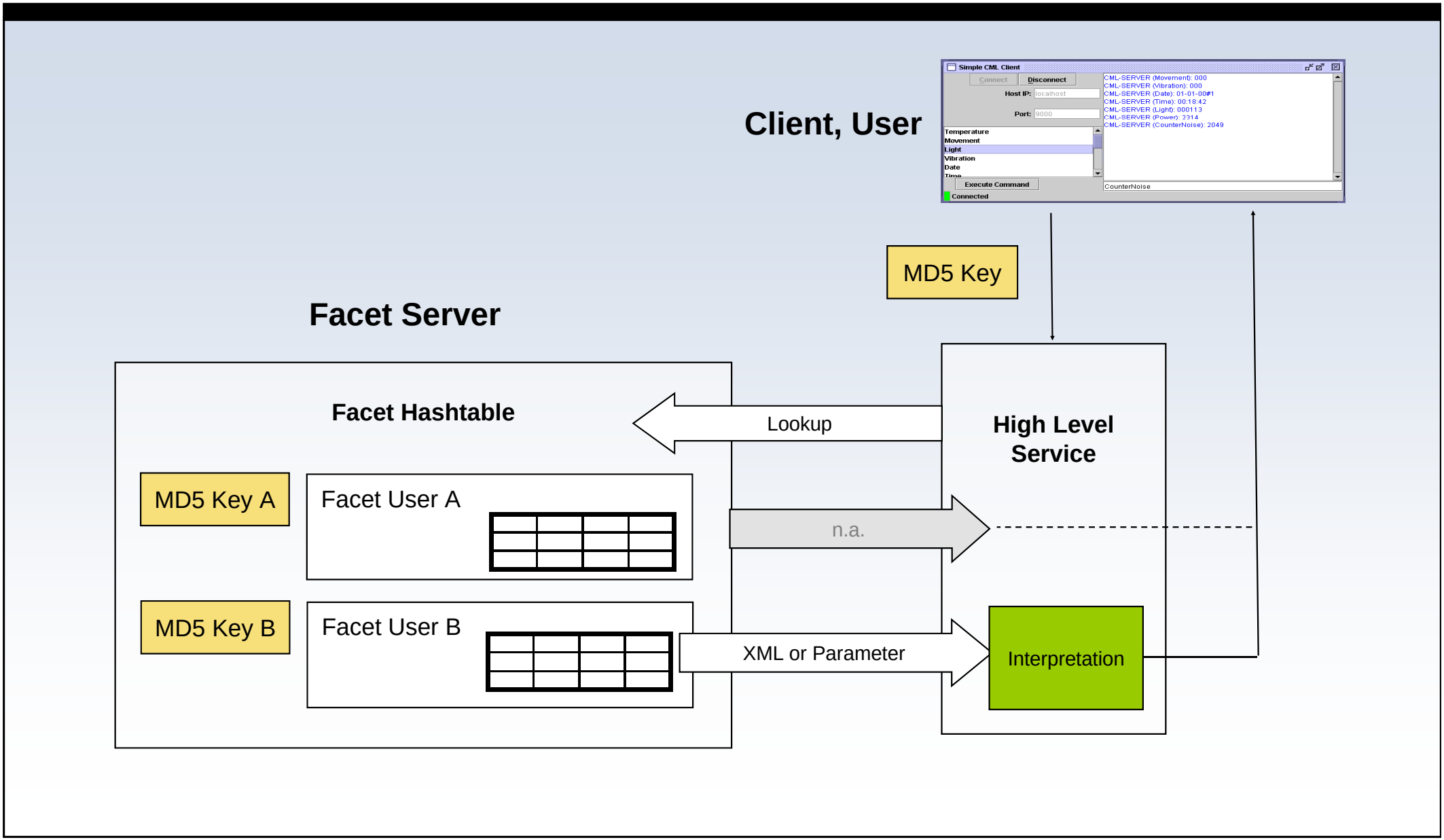
- Separation: Hardware ⇔ Sensor
- For each SensorID one single Sensor
- SensorID: unique, persistent, fixed to sensor
- HardwareID: ID for hardware implementations: ESB COM, ESB RF, ESB Socket
- ModuleID: ID for different hardware modules over the same hardware implementation: e.g. 3 ESB modules, one via COM, the two others via RF, each Module has different ModuleID, but they have the same HardwareID

- Services:
  - **Low Level:** Direct access to sensor values via Sensor ID, but hiding of all HardwareIDs or ModuleIDs and their methods
  - **High Level:** Aggregators of various Sensors, business logic, processing, interpretation

1. Security, MD5, Architecture



1. Security, MD5, Architecture



1. Login and Security

MD5 – Message Digest

MD5

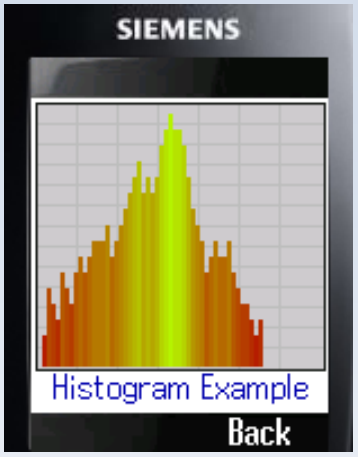
Start tool: 

2. Mobile Device 2D GraphClass

2. Mobile Device 2D GraphClass

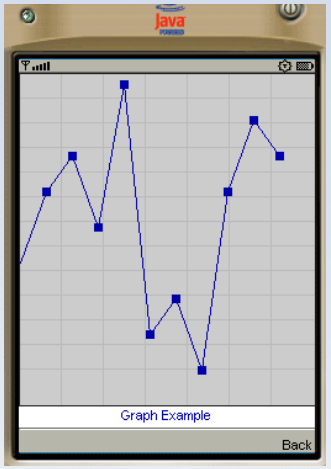
GraphBar

- 3 to 15 values
- Awareness the next hours
- Week overview (days)



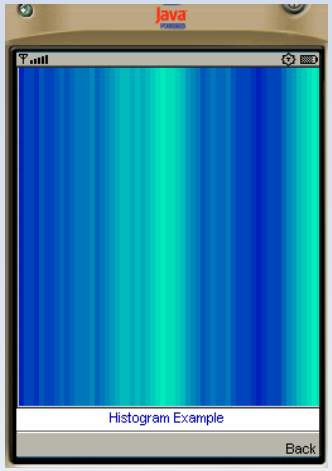
GraphHistogram

- 40 to 100 values
- More values with compression method
- Usage patterns visualization
- Room profile
- Day overview (high resolution)



GraphLine

- 3 to 20 values
- With or without bullet points
- ToDo: multiple lines



GraphGradient

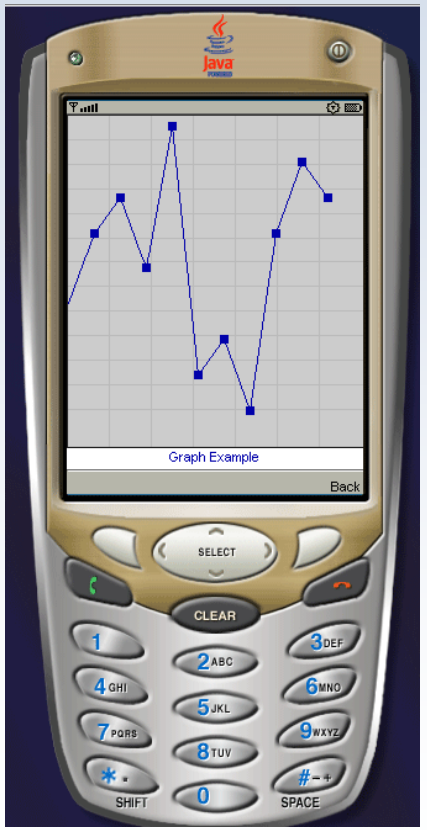
- Similar to GraphHistogram
- Combination with GraphLine possible

2. Mobile Device 2D GraphClass

GraphLine, GraphGradient:

```
if(size >= 2){
    int value = 0;
    int nextValue = 0;
    for (int i = 0; i < size - 1; ++i){
        value = ((Integer)values.elementAt(i)).intValue();
        nextValue = ((Integer)values.elementAt(i+1)).intValue();
        g.drawLine((horiBegin + i * barWidth),
                (vertEnd - value * barStepSize),
                (horiBegin + (i+1) * barWidth),
                (vertEnd - nextValue * barStepSize));

        g.fillRect((horiBegin + (i+1) * barWidth) - (bulletSize / 2),
           (vertEnd - nextValue * barStepSize) - (bulletSize / 2),
           (bulletSize),
           (bulletSize));
    }
}
```



2. Mobile Device 2D GraphClass

2D Engine, GraphLine, GraphGradient

Eclipse: J2ME Project

Start tool: 

3. Java Sound and Audio

3. Java Sound and Audio

J2SE Client "AudioAwareness":

- Audio playback is affected by awareness information
- Sensors:
  1. Noise, noise counter and average
  2. Movement detection
  3. (Light)
  4. (Vibration)
- We have to pay attention to the following cases:
  - Prevention of quick changes: 1 minute steps for decisions
  - Sound playback loops: continuous sound loops
  - Cross fading of the sound loops
  - Beware of feedback loops: especially when couple volume to noise
- Further aspects:
  - Loops of classical music: strings, piano to full orchestra (forte)
  - 5 Levels of activity
  - Begin with lowest level and collect information at least for 1 minute

3. Java Sound and Audio

Java Sound API:

- Sound device integration: channels, lines, streams, mixer
- MP3 Support: Tritonus Open Source Java Sound
  - Implementation of the Java Sound API, designed for i386 Linux
  - Additional generic pure Java plugins: MP3 decoder, encode, etc.
  - [http://tritonus.org/]
- MP3 Support: Javazoom MP3 SPI
  - http://www.javazoom.net/mp3spi/
- Development information about Java Sound API:
  - http://www.sun.com/j2se/1.4.2/docs/guide/sound/programmer_guide/
  - Java Sound Resources: http://www.jsresources.org/

- *Lack of MP3 support and no fading/mixing between samples,*
- *Also no DSP Support or audio effects*

3. Java Sound and Audio

LWJGL – Lightweight Java Game Library:

- Java implementation (or connector) for using OpenGL and OpenAL
- OpenAL – Open Audio Library:
  - Supports multiple channels
  - Mixing of audio streams
  - Audio positioning: Exact "virtual" locations of sounds
  - Multiple Digital Sound Processing effects
  - More information: http://www.openal.org
- Under BSD License, good developer documentation, FAQ
- Current release: LWJGL 0.93, http://www.lwjgl.org

→ Development of a real cross fading utility

- Audio buffers for 3 (or more) sound loops: classical music
- Fading process is executed by a thread
- Controls for fading up and down

3. Java Sound and Audio

CrossFading:

Libraries:

```
import org.lwjgl.BufferUtils;
import org.lwjgl.LWJGLException;
import org.lwjgl.Sys;
import org.lwjgl.openal.AL;
import org.lwjgl.openal.AL10;
import org.lwjgl.test.openal.WaveData;
```

Buffers:

```
/** Sources are points emitting sound. */
IntBuffer source = BufferUtils.createIntBuffer(NUM_BUFFERS);

/** Position of the source sound. */
FloatBuffer sourcePos = BufferUtils.createFloatBuffer(3 * NUM_BUFFERS);

/** Velocity of the source sound. */
FloatBuffer sourceVel = BufferUtils.createFloatBuffer(3 * NUM_BUFFERS);

/** Position of the listener. */
FloatBuffer listenerPos = BufferUtils.createFloatBuffer(3).put(
        new float[] { 0.0f, 0.0f, 0.0f });

/** Velocity of the listener. */
FloatBuffer listenerVel = BufferUtils.createFloatBuffer(3).put(
        new float[] { 0.0f, 0.0f, 0.0f });


/** Fading Thread */
private Thread thread = null;
```

3. Java Sound and Audio

Load music files:

```
waveFile = WaveData.create("PLoop3.wav");
AL10.alBufferData(buffer.get(MUSICFILE3), waveFile.format,
        waveFile.data, waveFile.samplerate);
waveFile.dispose();

AL10.alSourcei(source.get(MUSICFILE3), AL10.AL_BUFFER, buffer
        .get(MUSICFILE3));
AL10.alSourcef(source.get(MUSICFILE3), AL10.AL_PITCH, 1.0f);
AL10.alSourcef(source.get(MUSICFILE3), AL10.AL_GAIN, 1.0f);
AL10.alSource(source.get(MUSICFILE3), AL10.AL_POSITION,
        (FloatBuffer) sourcePos.position(MUSICFILE3 * 3));
AL10.alSource(source.get(MUSICFILE3), AL10.AL_VELOCITY,
        (FloatBuffer) sourceVel.position(MUSICFILE3 * 3));
AL10.alSourcei(source.get(MUSICFILE3), AL10.AL_LOOPING, AL10.AL_TRUE);
```

Start playing loops, and change gain settings:

```
setListenerValues();

AL10.alSourcePlay(source.get(MUSICFILE1));
AL10.alSourcePlay(source.get(MUSICFILE2));
AL10.alSourcePlay(source.get(MUSICFILE3));

AL10.alSourcef(source.get(MUSICFILE1), AL10.AL_GAIN, 1.0f);
AL10.alSourcef(source.get(MUSICFILE2), AL10.AL_GAIN, 0.0f);
AL10.alSourcef(source.get(MUSICFILE3), AL10.AL_GAIN, 0.0f);
```

3. Java Sound and Audio

Fading thread:

```
public void run() {
    while (true) {
        try {
            fader += fadeStep;

            if (fader >= 1.0f) {
                // Last fading step: maximum/minimum values for the channels
                AL10.alSourcef(source.get(fadeFrom), AL10.AL_GAIN, 0.0f);
                AL10.alSourcef(source.get(fadeTo), AL10.AL_GAIN, 1.0f);
                currentlyFading = false;
                thread.stop();
            } else {
                // Gain control for the two channels
                AL10.alSourcef(source.get(fadeFrom), AL10.AL_GAIN,
                        (1.0f - fader));
                AL10.alSourcef(source.get(fadeTo), AL10.AL_GAIN,
                        (0.0f + fader));
            }
            Thread.sleep(500);
        } catch (InterruptedException e) {
            e.printStackTrace();
            currentlyFading = false;
        }
    }
}
```

File Playback, MP3, CrossFading Utility

Ecplise: JavaSoundMP3

Start tool: 

4. Bluetooth

4. Bluetooth

- Bluetooth development resources:
  - Many developer informations and tools: http://www.benhui.net
  - Software overview: http://www.javabluetooth.com/software.html
  - Article: http://today.java.net/pub/a/today/2004/07/27/bluetooth.html
- Bluetooth and Mac OS X:
  - No free available Bluetooth JSR-82 implementation for OS X
  - Commercial product: AvetanaBluetooth http://wwwavetana-gmbh.de
  - Alternative: Windows PC as Proxy
- Windows XP:
  - Microsoft Bluetooth stack has to be installed: only available via XP SP 2
  - Bluetooth dongle or integrated device needed
  - BlueCove Stack needed (is a implementation on top of the Microsoft API): http://sourceforge.net/projects/bluecove
  - Stack: serve as a bridge between Java programs and Bluetooth hardware

Thank You

For Your Attention!