Java™ Platform, Micro Edition

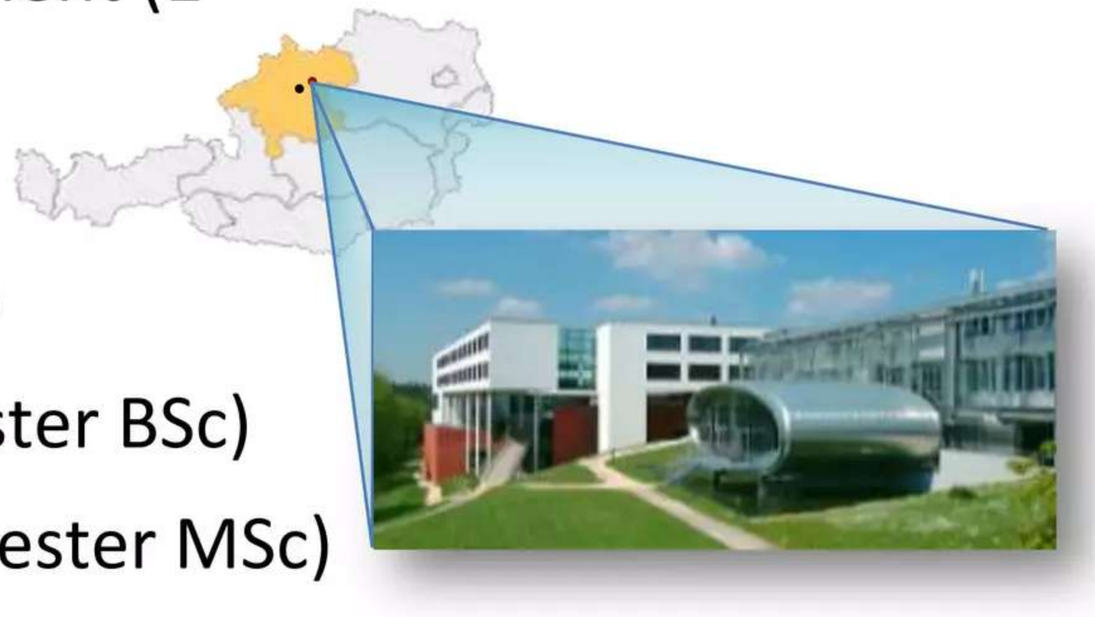
Part 1 – Introduction to Java ME, CLDC and MIDP

Disclaimer

- **These slides are provided free of charge at http://www.symbianresources.com and are used during Java ME courses at the University of Applied Sciences in Hagenberg, Austria at the Mobile Computing department (http://www.fh-ooe.at/mc)**
- **Respecting the copyright laws, you are allowed to use them:**
  - **for your own, personal, non-commercial use**
  - **in the academic environment**
- **In all other cases (e.g. for commercial training), please contact andreas.jakl@fh-hagenberg.at**
- The correctness of the contents of these materials cannot be guaranteed. Andreas Jakl is not liable for incorrect information or damage that may arise from using the materials.
- This document contains copyright materials which are proprietary to Sun or various mobile device manufacturers, including Nokia, SonyEricsson and Motorola. Sun, Sun Microsystems, the Sun Logo and the Java™ Platform, Micro Edition are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

About me: Andreas Jakl

- Assistant Professor at the **University of Applied Sciences, Hagenberg** since 2006
- **Teaching:**
  - Introduction to Software Development (1st semester BSc)
  - **Java ME (2nd semester BSc)**
  - Qt / Symbian OS (3rd semester BSc)
  - Bachelor Thesis Seminar (5th semester BSc)
  - Mobile Operating Systems (1st semester MSc)
  - Interaction Technology (2nd semester MSc)



About me: Andreas Jakl

- **Experience:**
  - **Forum Nokia Champion** (2007, 2008, 2009)
  - Founded company "**Mopius**" in 2004
  - Internship, Master's Thesis and summer jobs at **Siemens / BenQ Mobile** (Munich, R&D)
  - Studied Bachelor & Master of **Digital Media** in Hagenberg / Austria (2001 – 06)
- **Contact:**
  - Office A.005a (FH1, lower floor)
  - andreas.jakl@fh-hagenberg.at

What is it all about?

Java Platform

The Java Platform, Part 1 / 3

- **Java programming language**
- Compared to C++:
  - No pointers
  - Automatic garbage collection
  - Interfaces instead of multiple inheritance
  - Comes with an extensive library

The Java Platform, Part 2 / 3

- **Virtual Machine (JVM)**
  - Executes compiled Java bytecode (.class)
  - Available for many systems
  - Controls the code (security)

- Often used for mobile phones:
  Kilobyte Virtual Machine (KVM)
  - Memory footprint starting at only 60 kB (+)

- Now being replaced by Hotspot JVMs.

The Java Platform, Part 3 / 3

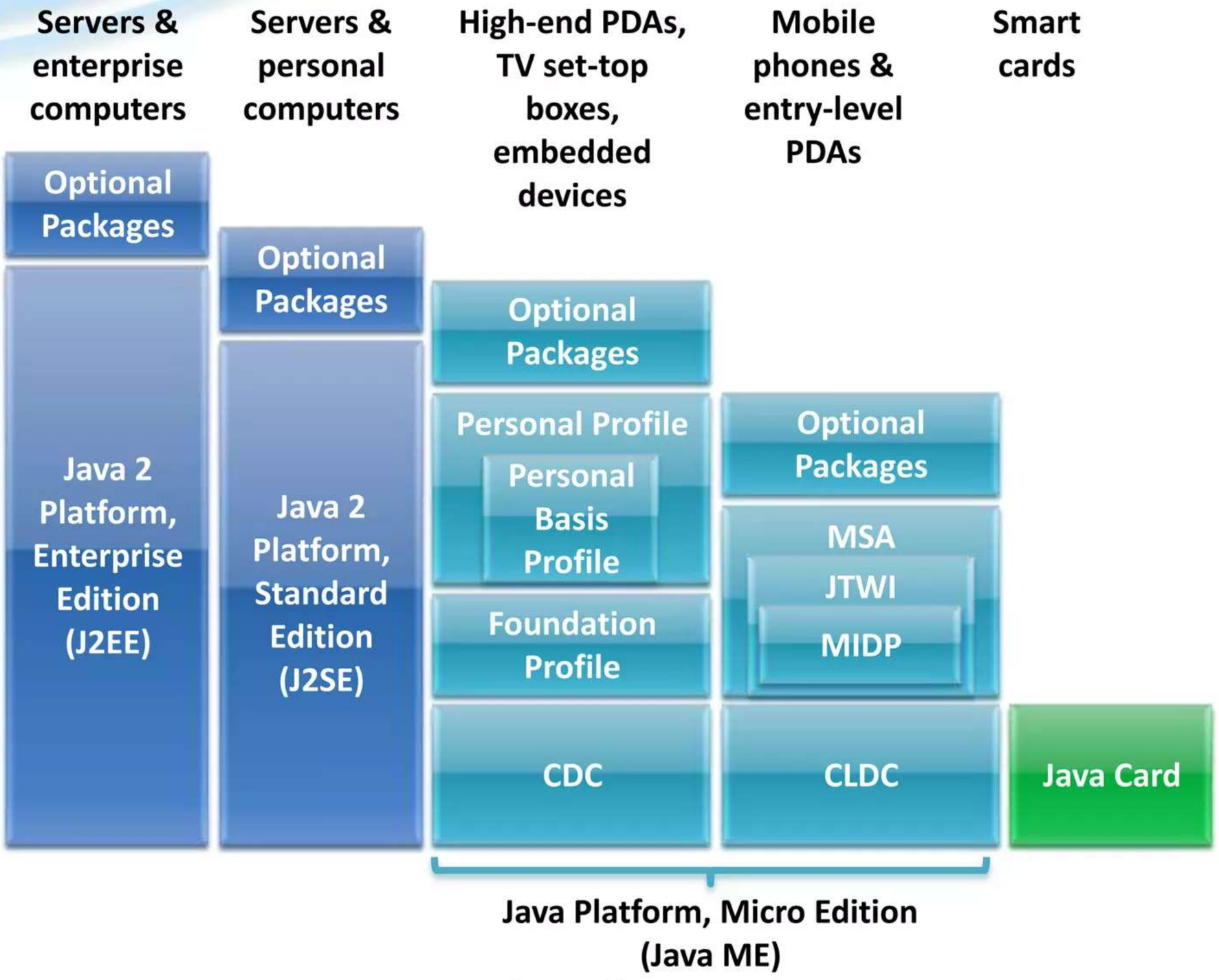
- **Application Programming Interfaces (APIs)**
  - Manifold libraries
  - e.g. UI, network, 3D, location based services, etc.

Java?

- **In numbers:**
  - > 6 Billion Java-enabled devices
  - **2.6 Billion Java-enabled phones (8 out of 10 shipped in 2008)**
  - 3.5 Billion Java Cards
  - 20 Million Java set-top boxes
  - 800 Million Java desktops
  - **180 Operators deploying Java content**
  - 6 Million developers

Java Editions

MSA … Mobile Service Architecture (JSRs 248 and 249)
JTWI … Java Technology for the Wireless Industry, JSR 185
MIDP … Mobile Information Device Profile
CDC … Connected Device Configuration
CLDC … Connected Limited Device Configuration



Going Mobile

Java ME

Differences J2SE / Java ME (MIDP)

- Java ME is mainly a **subset of J2SE**
  - But **different UI- and event handling** functionality
  - **Less utility classes**
    (only Vector and Hashtable, no LinkedLists, …)
- **Code runs on both platforms?**
  - → general algorithms: yes
  - But the whole application needs porting

Name: J2ME or Java ME?

- **Official name:**
  - Java Platform, Micro Edition (Java ME)
- **Former name:**
  - J2ME

Configuration

- Defines Java Platform for different device classes:
- **CLDC**
  - Limited user interface
  - Low computing power (usually with a battery)
  - Network with low bandwidth
- **CDC**
  - Network connection with high bandwidth, possibly persistent
  - Larger memory requirements

CDC

- **Equivalent to Java SE 1.4.2 when combined with:**
  - **Foundation Profile** (FP)
    - Extends CDC to Java SE 1.4.2, without graphics and UI
  - **Personal Basis Profile** (PBP)
    - Lightweight GUI support (AWT subset)
  - **Personal Profile** (PP)
    - Extends PBP with AWT components and Appled support
- **Foundation for Java-based platforms:**
  - (few) smartphones, Blu-Ray, Set-top boxes, etc.

CLDC

- Currently available in two versions:
- **1.0**
  - In devices until ~ 2005 (at the latest)
- **1.1**
  - Current standard
  - Supports floating point
    (but mostly in software → slow)
  - Important e.g. for GPS coordinate handling!
- **Used for:**
  - Phones (!)
  - Consumer and entertainment devices
  - Embedded platforms, controllers, sensors
  - Sun SPOTs

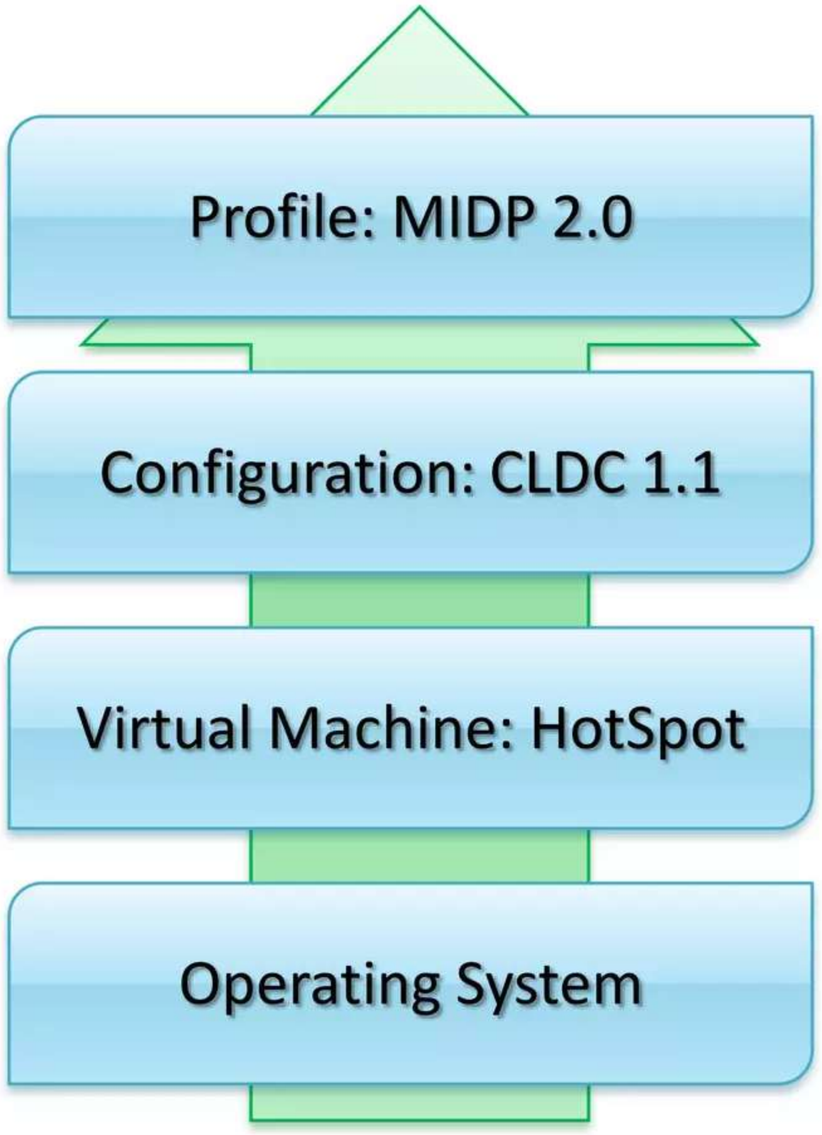
Profiles

- Extension and more detailed specification for a configuration
- Contains **APIs** for UI, event handling, data storage, networks, timers, …
- **Minimum requirements** for devices (screen size, input possibilities, ...)
- For mobile phones:
  Mobile Information Device Profile (MIDP)

Profiles – Major Differences

- **MIDP 1.0**
  - Hardly any sound support, limited graphics
  - Only HTTP, no Sockets
  - → Many vendor-specific extensions (bad!)
- **MIDP 2.0**
  - Game API
  - Better network and multimedia support
- **MIDP 2.1**
  - Tries to improve fragmentation problems (different version for every phone…)
  - Stricter specifications for packages
- **MIDP 3.0 (JSR 271)**
  - Multiple MIDlets can run concurrently or in the background, auto-launch
  - More detailed specifications
  - More possibilities for the UI, support of secondary displays

Sample Architecture of a Phone



Java ME – Examples

- **Motorola MOTORAZR V3**
  - CLDC 1.0
  - MIDP 2.0

Java ME – Examples

- **Nokia N70**
  - CLDC 1.1
  - MIDP 2.0

Java ME – Examples

- **SonyEricsson P990i**
  - CLDC 1.1
  - MIDP 2.0
  - CDC 1.0
  - Personal Profile

Java ME – Examples

- **Nokia N86 8MP**
  - CLDC 1.1
  - MIDP 2.1
  - MSA (Subset)

Java ME – Examples

- **Amazon Kindle 2**
  - CDC

JSRs (Java Specification Requests)

- JSR = CLDC, MIDP or libraries for additional features
- Defined through:
  Java Community Process (JCP)
- Examples:
  - JSR 82: Bluetooth APIs
  - JSR 179: Location API
  - JSR 184: Mobile 3D API
  - JSR 226: Scalable 2D Vector Graphics API

JSRs – How Many?

- **Sample:** supported JSRs of Nokia N86 8MP

Java Technology	JSR 139 Connected, Limited Device Configuration (CLDC) 1.1 JSR 118 MIDP 2.1 JSR 248 Mobile Service Architecture Subset for CLDC JSR 75 FileConnection and PIM API JSR 82 Java™ APIs for Bluetooth 1.1 JSR 135 Mobile Media API 1.1 JSR 172 J2ME™ Web Services Specification (RPC package) JSR 172 J2ME™ Web Services Specification (XML Parser package) JSR 177 Security and Trust Services API for J2ME™ (SATSA-CRYPTO package) JSR 177 Security and Trust Services API for J2ME™ (SATSA-PKI package) JSR 179 Location API for J2ME™ 1.0 JSR 180 SIP API for J2ME™ JSR 184 Mobile 3D Graphics API for J2ME™ 1.1 JSR 205 Wireless Messaging API 2.0 JSR 226 Scalable 2D Vector Graphics API for J2ME™ 1.1 JSR 234 Advanced Multimedia Supplements 1.0 (audio3d) JSR 234 Advanced Multimedia Supplements 1.0 (music) IAP Info API eSWT UI API 1.0.3 Nokia UI API 1.1

JSRs – Games?

Asphalt 4: Elite Racing HD
© Gameloft

- A racing game could require:
  - **JSR 184 (3D Graphics)**
    - 3D world
  - **JSR 135 (Mobile Media)**
    - Sound
  - **JSR 82 (Bluetooth)**
    - P2P Gaming
  - **JSR 180 (SIP)**
    - P2P over the network
  - **JSR 229 (Payment)**
    - New forms of payment

JSRs – Mapping Applications?

Google Maps Mobile
© Google

- A mapping application could require:
  - **JSR 226 (Vector Graphics)**
    - Map visualization
  - **JSR 179 (Location)**
    - Where am I?
  - **JSR 172 (Web Services)**
    - Requesting data
  - **JSR 75 (File and PIM)**
    - Map an address
  - **JSR 238 (Internationalization)**
    - Global software

„Write Once, Run Anywhere™“ ?

- **Problems:**
  - Different screen sizes
  - Bugs in manufacturers implementations (!)
  - Different hardware performance
  - Which JSRs are supported? Bluetooth? SVG? Web services?…
  - MIDP 2.0 isn’t strict enough:
    - Different key codes for every manufacturer (softkeys, ...)
    - Are socket connections available?
    - Which sound files are playable? Supported image formats?
    - Is double buffering supported?

JTWI – Fragmentation Solution?

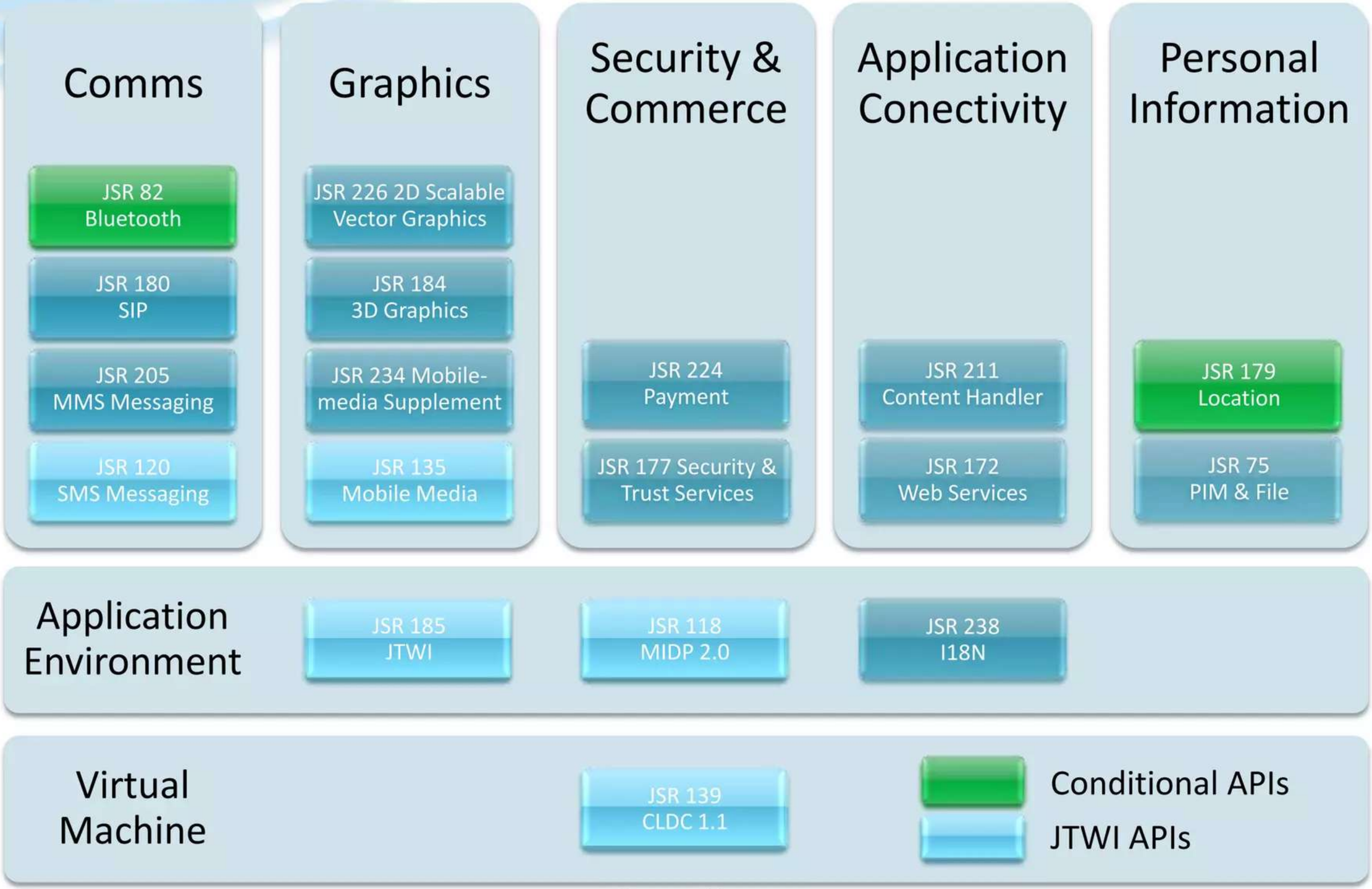
- **Java Technology for the Wireless Industry (JSR 185)**
  - First try of defining full API stack to reduce fragmentation
  - Clarification of component JSR specifications
  - **However:** too small, too few APIs included
  - Bad licensing politics

→ **Failed**

Mobile Service Architecture (MSA, JSR 248)

- **Goal?**
  - "Umbrella" specification, replace JTWI
  - Define a **unified platform** for majority of handsets
    - Spec leds: Nokia, Vodafone. Others heavily involved
- **Devices started to ship in 2007**
  - With at least subset of MSA

Features in MSA for CLDC



MIDP 3.0

- **High-level goals**
  - Add much-requested **functionality over MIDP 2**
  - Rework security model to support CLDC and CDC
    - Enables support of **MIDP 3 on CDC**
  - **Clarify spec** and increase **implementation consistency**

MIDP 3.0 Functional Enhancements

- Concurrency (Multiple MIDlets at the same time)
- Shared libraries (LIBlets)
- Auto-start MIDlets
- Idle screen MIDlets
- Inter-MIDlet communication
- Record store interchange format
- User Interface improvements
- ...

Mobile Service Architecture v2

- **Dynamic Environment**
  - Download new APIs to the handset
  - Place custom middleware on handset
- **Next revision of MSA 248**
  - Supports both CLDC and CDC
  - Builds upon and requires MIDP 3.0
  - Adds multi-tasking and on-device service framework
  - Adds competitive user interface toolkit
  - Adds device segments: entry, standard, advanced
- **No final release date yet**

JavaFX

- **JavaFX** (http://javafx.com – integrated in NetBeans 6.5+)
  - New **UI libraries** (graphics, media, web services)
  - **Consistent experience** across mobile, desktop, browser, TV, etc
  - **Plus:** use any Java library in JavaFX
  - Integrated with Java Runtime
- **JavaFX Script**
  - Simple declarative language, easier to learn
  - e.g., for artists to change sprite animation, without needing software developer
  - Advantage to JavaScript / ActionScript: integration with Java – reuse any Java library

JavaFX Mobile

- **Runs on Java ME (plus Android)**
  - Mobile content with same tools as Java FX
- **Availability?**
  - JavaFX Mobile Runtime needs to be pre-installed on the phone
  - No phones released yet
  - Currently endorsed by: SonyEricsson, LG

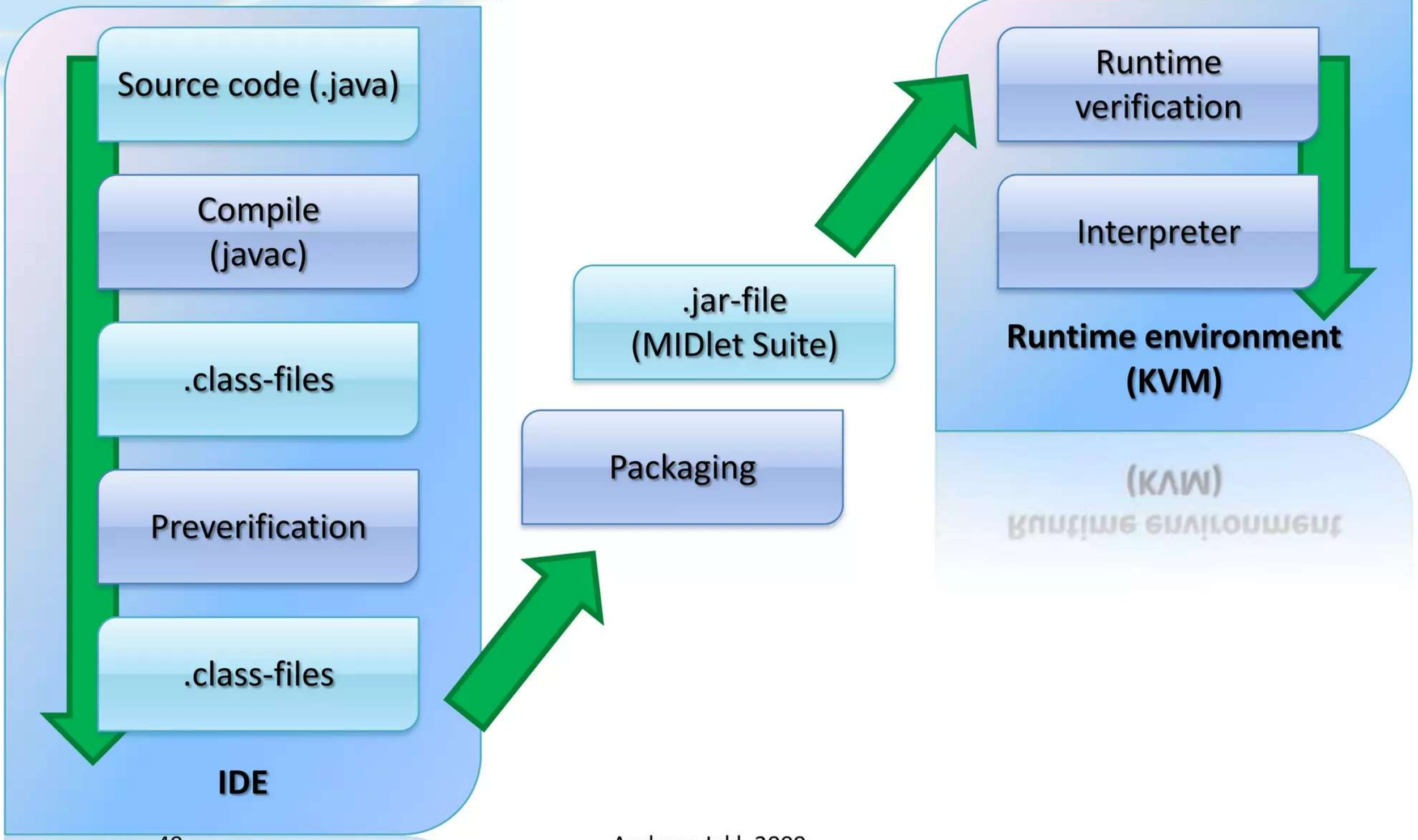
Blu-Ray Disc Java: BD-J

- Based on **Java ME Personal Basis Profile** & earlier Java TV spec
  - GUI environment suitable for consumer electronics (no keyboard / mouse)
  - Strong media support
- **Applications?**
  - Dynamic menu systems
  - Downloading additional content (subtitles, movie previews, etc.)
  - Games and other bonus material

Internals

Java ME – Applications

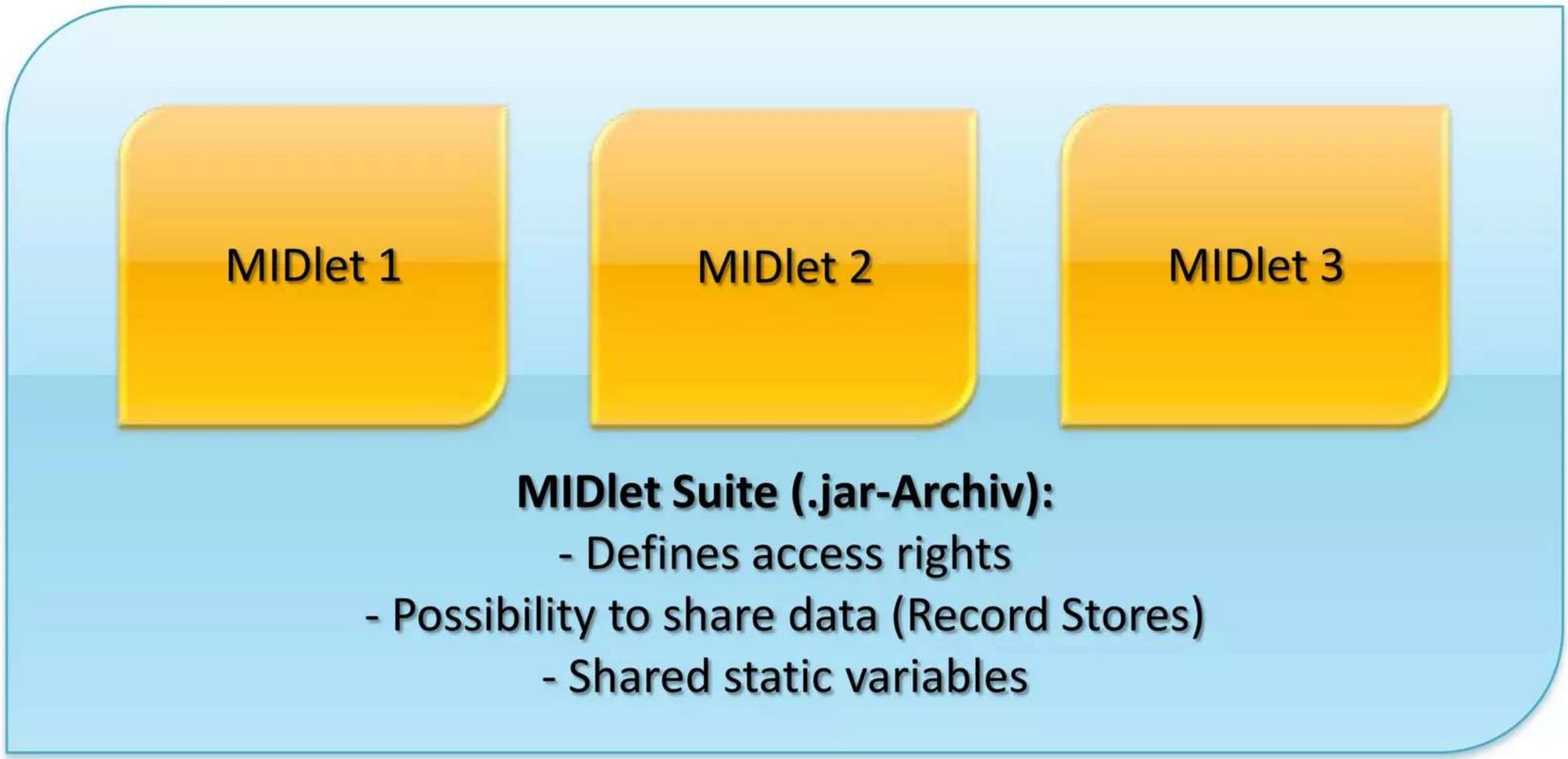
Code → Application



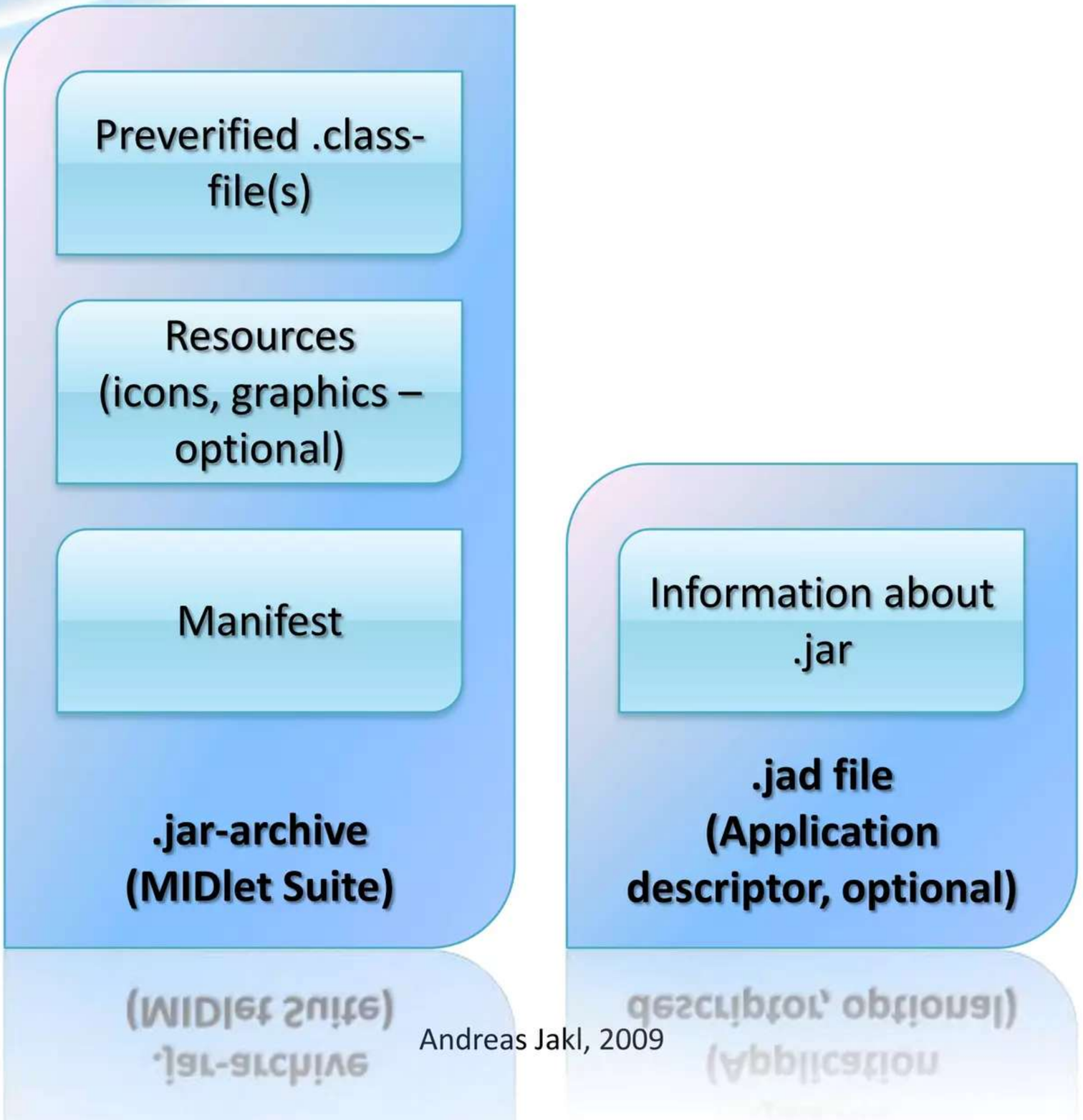
Preverification

- **Verification**: check the integrity of the byte code at runtime
- → Too much for mobile devices, therefore: **pre-verification at compile time:**
  - Takes care of resource demanding checks
  - Simplifies runtime verification
  - Adds additional attributes to the .class file (5 – 15% increase in size)

MIDlet Suite



MIDlet-Suite



Manifest

- Text file “MANIFEST.MF”
- Contains meta information

```
Manifest-Version: 1.0
MIDlet-Name: RealReplay
MIDlet-Description: RealReplay
MIDlet-Vendor: Mopius
MIDlet-Info-URL: http://www.mopius.com/
MIDlet-Version: 0.96.20
MIDlet-Icon: /res/icon.png
MIDlet-1:
RealReplay,/res/icon.png,com.mopius.realreplay.RealReplayMIDlet
MicroEdition-Profile: MIDP-2.0
MicroEdition-Configuration: CLDC-1.1
```

Application Descriptor (.jad)

- Information about .jar contents
- Allows to check compatibility before downloading .jar

```
MIDlet-Name: RealReplay
MIDlet-Description: RealReplay
MIDlet-Vendor: Mopius
MIDlet-Info-URL: http://www.mopius.com/
MIDlet-Version: 0.96.20
MIDlet-Icon: /res/icon.png
MIDlet-1:
RealReplay,/res/icon.png,com.mopius.realreplay.RealReplayMIDlet
MicroEdition-Profile: MIDP-2.0
MicroEdition-Configuration: CLDC-1.1
MIDlet-Jar-Size: 114185
MIDlet-Jar-URL: http://realreplay.mopius.com/files/realreplay.jar
```

Add.

Signing

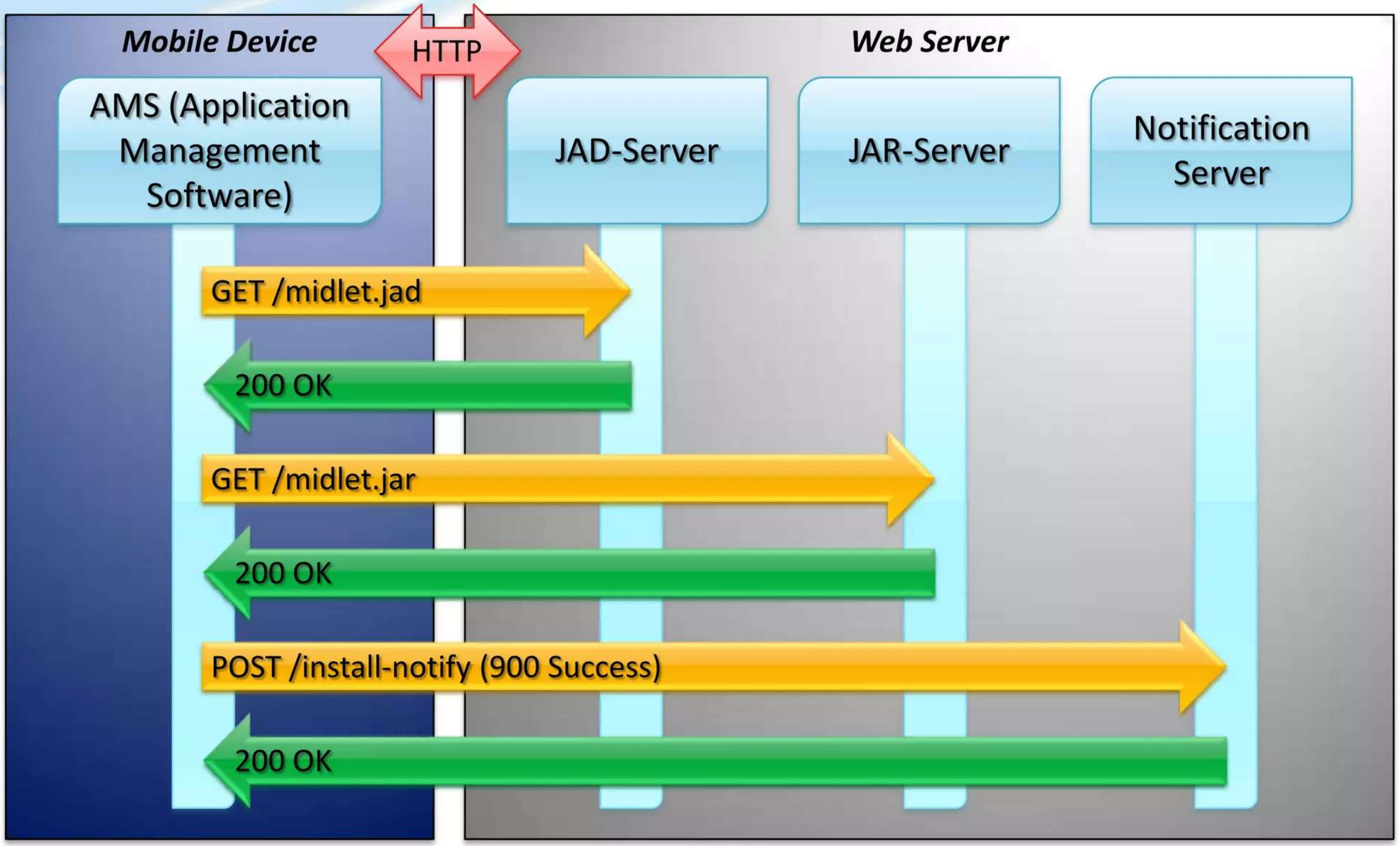
- Access to some telephone functionality (e.g. socket connections, SMS) restricted: warning is displayed every time
- **Solution:**
  - Sign the checksum of .jar in .jad-file with an own key (certificate)
  - Own certificate is signed with root certificate of a trusted certificate authority

Why OTA for Deployment?

- Some phones (Samsung, Sagem, BREW,...) do not support installing MIDlets through the PC or Bluetooth
- **Only alternative:**
  - Download directly through mobile phone

    → Over-the-Air (OTA) delivery

Over-the-Air


Mobile Device
HTTP
Web Server
AMS (Application Management Software)
JAD-Server
JAR-Server
Notification Server
GET /midlet.jad
200 OK
GET /midlet.jar
200 OK
POST /install-notify (900 Success)
200 OK

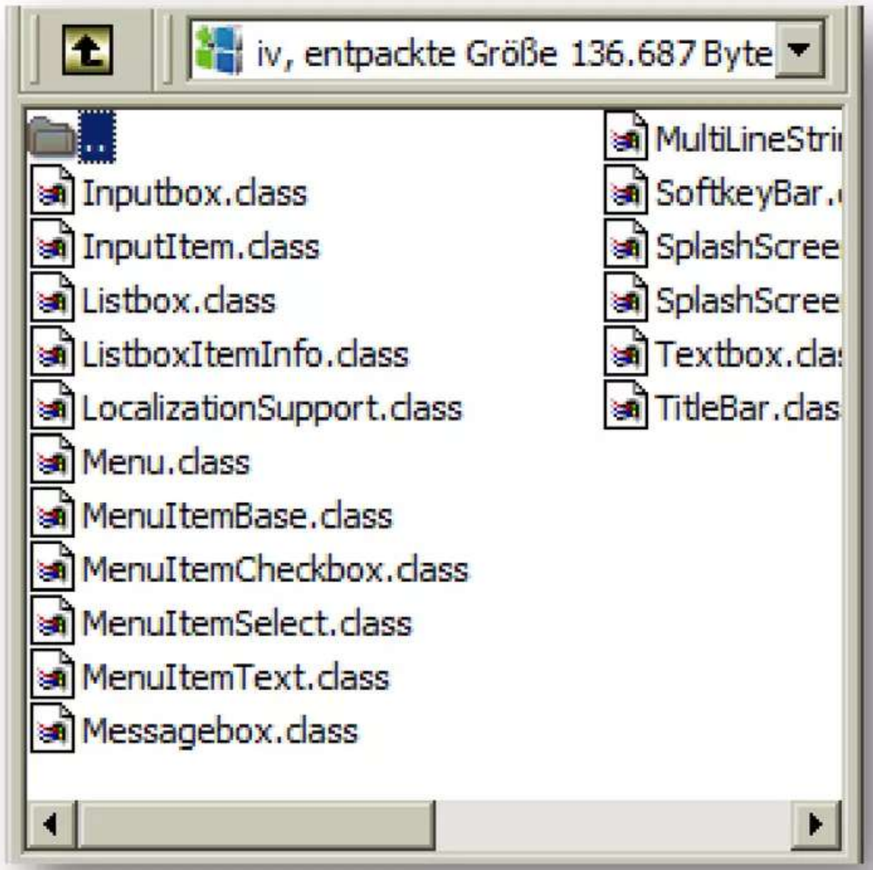
Optimization – Obfuscation

- **Original intention:**
  - Make reverse engineering more difficult
  - Code more difficult to read after de-compilation
- Renames classes to “a.class, b.class, …”
- Removes unused methods, variables, classes
- → **Significant size reduction**
  - Over-the-Air = expensive!
  - MIDlet size restrictions in many phones
  - Improves speed (less code to load / parse)

Obfuscation

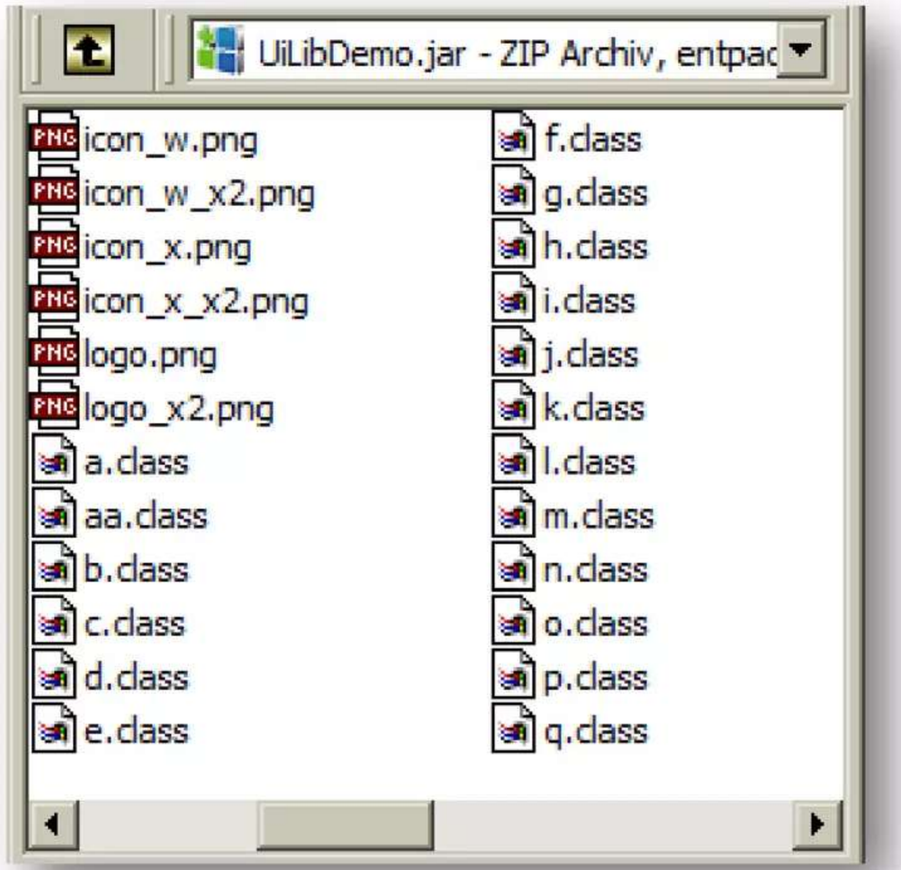
Original archive

79,2 kB



Obfuscated

42,9 kB = 45% smaller!



Developing for JavaME

- **Highest priority:** memory usage and speed
- → often very few classes, object orientation reduced to a minimum, frequent use of static variables
- **But:** today's phones have got more memory

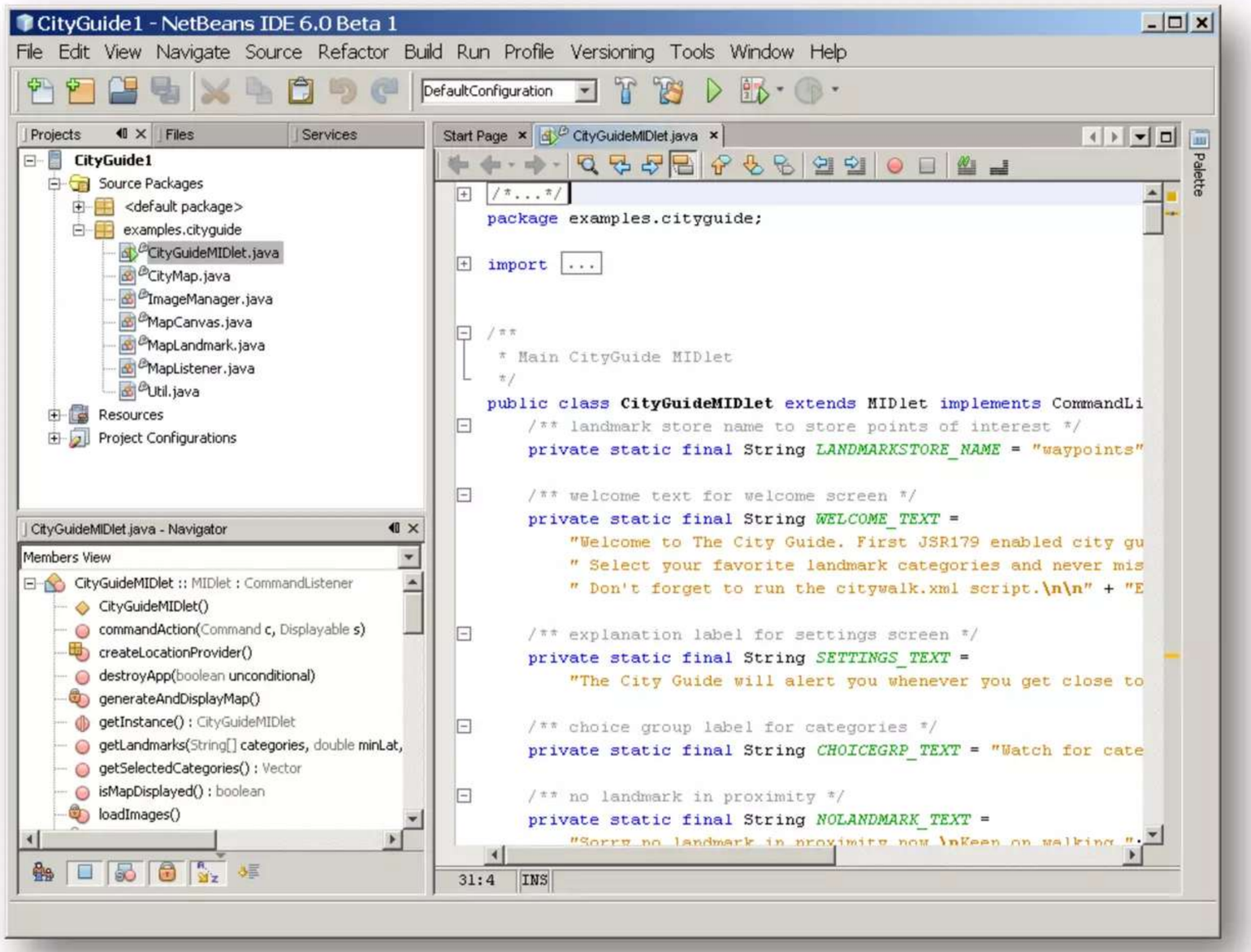
Example:
Winter Sports
from Digiment
Open Source (GPL)

How to develop:

Tools

IDEs: Sun NetBeans

- NetBeans (+ Mobility Pack)



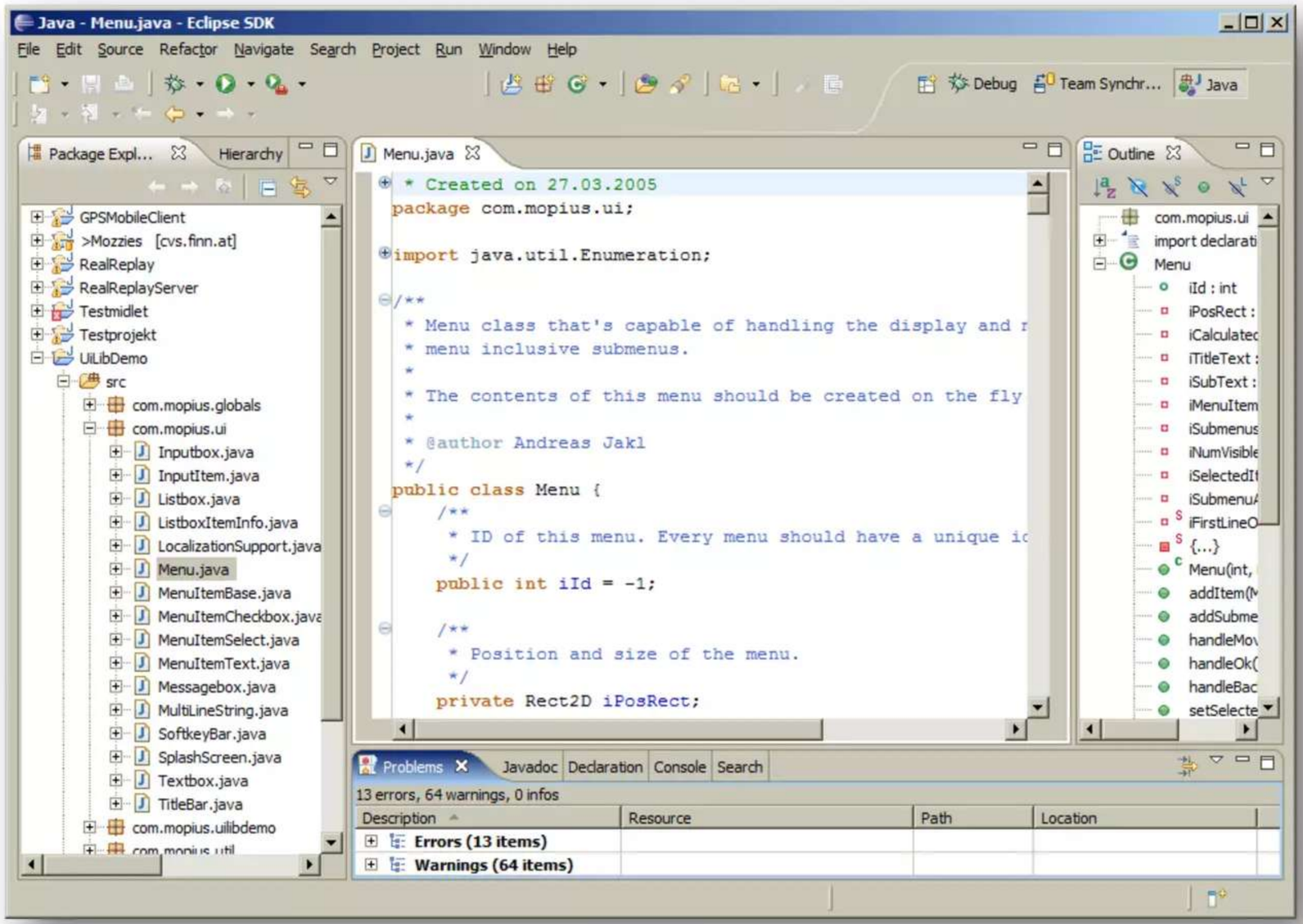
IDEs: NetBeans

- Very good integration for mobile projects:
  - Localisation
  - Conditional compilation
  - Packaging
  - UI-Designer, game builder, …

- Relatively high system requirements

IDEs: Eclipse

- Eclipse + Plugin: EclipseME / MTJ (official plug-in)



IDEs: Eclipse

- Very good IDE
- Helps a lot with programming errors

- ME integration through Eclipse ME is average
- Few Java ME specific tools / support – just pure Java

Emulators

- **Sun Java Wireless Toolkit for CLDC** (2.5.2 / 3.0 EA)
  - Tools for compiling, packaging and executing
- **Emulator:**
  - Debugging
  - Error handling
  - Text output through console
  - Allows performance analysis
  - Simulates internet access and GPS
- **Download:**
  - http://java.sun.com/javame/downloads/index.jsp

Sun WTK

Sample application of NetBeans in the WTK emulator

Manufacturer-Specific Emulators

- **Different Java ME implementations**
- → Every device manufacturer has its **own emulator**:
  - **Nokia:**
    www.forum.nokia.com/Resources_and_Information/Tools/IDEs/
  - **Sony Ericsson** SDK for the Java ME Platform
    developer.sonyericsson.com/site/global/docstools/java/p_java.jsp
  - **Samsung**
    innovator.samsungmobile.com/
  - **Motorola**
    developer.motorola.com/docstools/sdks/

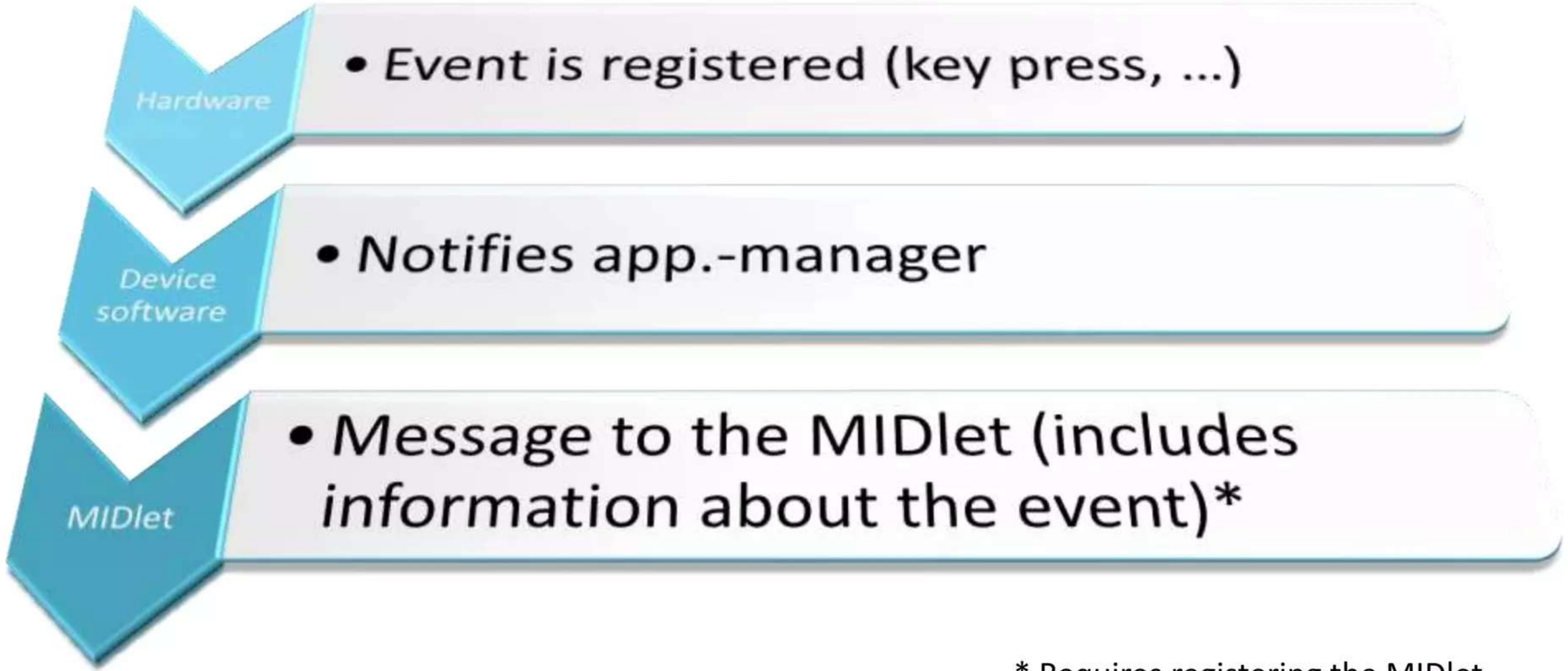
J2ME Polish

- **Tools suite** to address Java ME shortcomings:
  - Own **UI classes** for custom, graphical UI
  - **Build system**, creates adapted version for specific handsets
  - Allows porting Java ME to Android
- **Licensing:**
  - Free for GPL products
  - 1 commercial app: €990
  - Unlimited commercial apps: €14,990
- http://www.j2mepolish.org/

Commands

Event-Handling

Events – The Big Picture



* Requires registering the MIDlet

Listener

- Implement the **Listener-Interface** to get informed:
  - **CommandListener: commandAction()**
    Notification when e.g. a menu item has been selected
  - **ItemCommandListener: commandAction()**
    Used for events for individual items
  - **ItemStateListener: itemStateChanged()**
    When an UI element has been changed

Recap: Interfaces?

- **Interface: implemented by 1+ classes**
  - **Defines abstract methods** and constants
  - But **doesn't** contain the **implementation**!
  - Implementing class has to override all defined methods
  - Important for generic development!
  - Caller does not need to know exact class type and name, but can work with the interface type
- More information at: http://en.wikipedia.org/wiki/Interface_(Java)

Interface definition:

```
public interface Predator {
   boolean chasePrey(Prey p);
   void eatPrey(Prey p);
}
```

Interface implementation:

```
public class Cat implements Predator {
   public boolean chasePrey(Prey p) {
      // programming to chase prey p (specifically for a cat)
   }
   public void eatPrey (Prey p) {
      // programming to eat prey p (specifically for a cat)
   }
}
```

Commands

- Command = **semantic information** about an action (→ how can an action be executed?)
- But no actual implementation of the action!
- **Contains:**
  - Short label
  - Long label (optional)

  } One of them will be displayed on the screen / in the menu, depending on the available space

  - Type — "Intention" of the command – e.g. for special placement on the device
  - Priority — For the order of commands, if more are mapped to the same softkey. The lower the priority, the more important it is.

Command-Types

Type	Description
BACK	Navigate to the previous screen (logically)
OK	Standard positive answer
CANCEL	Standard negative answer, e.g. used for OK & CANCEL
EXIT	Exit the application
SCREEN	Application specific command, e.g. “Upload”
HELP	Request display of help text
ITEM	Command is specific for items of the `Screen` or for elements of a `Choice` component
STOP	Stop currently running process / task

Exit

Implementation
e.g. like in the
“Hello World”-
Exercise

- **Class `HelloWorldMIDlet`:**
  - `… implements CommandListener`
- **Define new command (member variable):**
  - `private Command cmdExit;`
- **Create it in the constructor:**
  - ```
    cmdExit = new Command(“Exit”, Command.EXIT, 1);
    frmMain.addCommand(cmdExit);
    frmMain.setCommandListener(this);
    ```
    ```

Command Handling

- Method defined in the base class `CommandListener`:

```
- public void commandAction (Command c, Displayable d)
  {

      if (c == cmdExit)
      {
          destroyApp(true);
          notifyDestroyed();
      }

  }
```

true: forces shutdown, you have to free resources!
false: shtudown can be prevented by the MIDlet if necessary.
... mainly important if the framework wants / has to close down the app.

Inform the app. manager that our MIDlet wants to be shut down.

Exit Softkey

2 Softkeys, 3 Commands?

- Commands in the menu – placement depends on the phone:

WTK 2.5

Nokia Series 40

Nokia S60


```
cmdExit = new Command("Exit", "Exit", Command.EXIT, 1);
cmdStart = new Command("Start", "Start Game", Command.SCREEN, 2);
cmdResume = new Command("Resume", "Resume Game", Command.SCREEN, 3);
```

That’s it!

Thanks for your attention