J2ME – Java-Technologie für mobile Endgeräte

Christian Eisoldt

Java Systems Engineer DSC

Sun Microsystems GmbH

J2ME: Java 2 MicroEdition

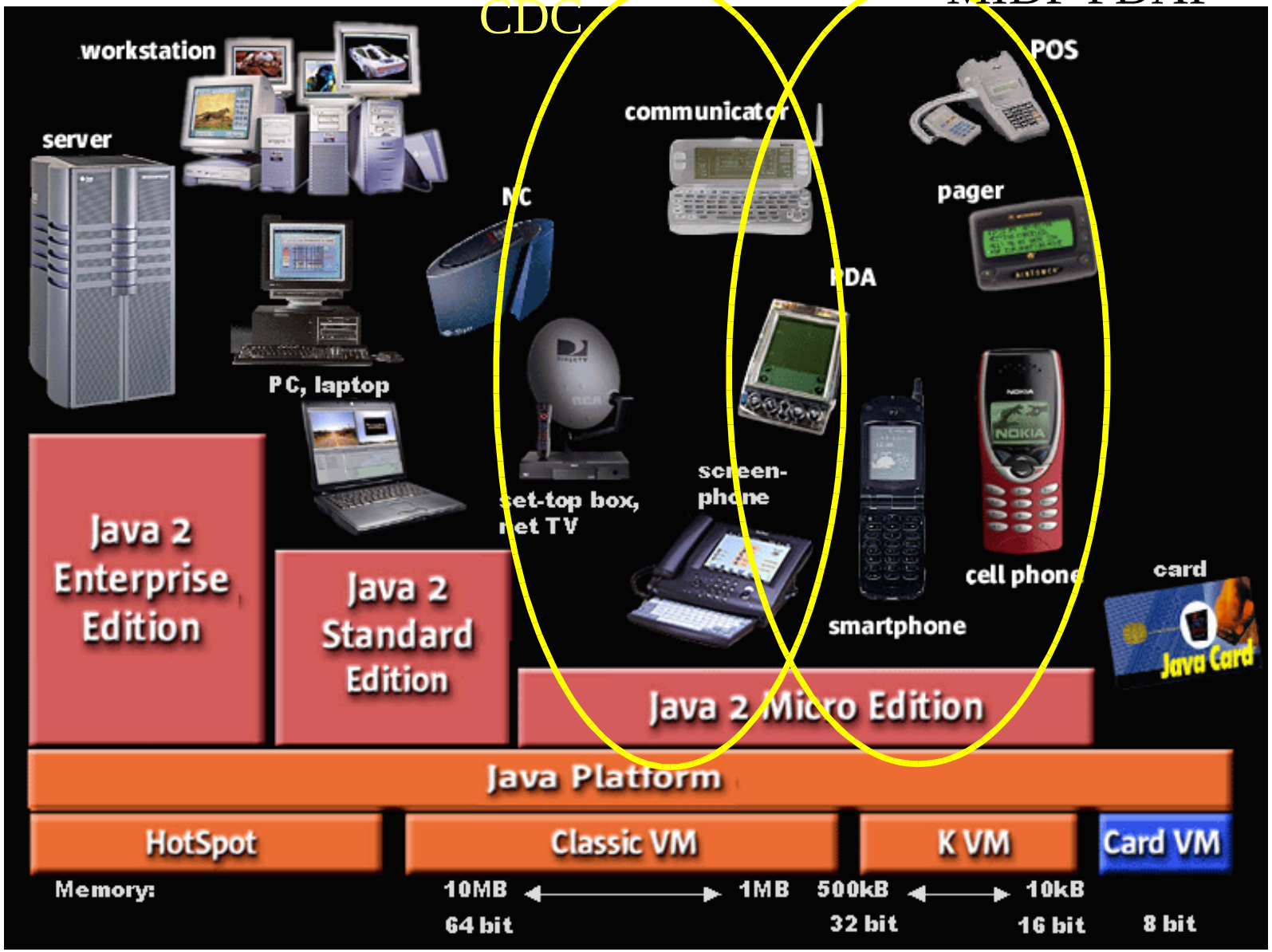
Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- J2ME und Webservices

Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- J2ME und Webservices

Java Skalierbarkeit



Java als portable Programmierplatform

- Eingeschränkte Rechenleistung
- Unterschiedliche Gerätetypen und Benutzerschnittstellen
- Inhalte werden von Service Provider bereitgestellt

Java VM für embedded Systeme?

- Normale (Desktop- und Server-) Java VMs eignen sich nicht für Embedded- und Consumer-Systeme
  - Statischer und dynamischer Memory-Bedarf zu groß
  - Nicht „ROMable“
  - Nicht hinreichend einfach portierbar (zu viele Assembler-Anteile nötig)
  - Nicht genügend modular
  - Kein deterministisches Verhalten (nicht RTOS aware)

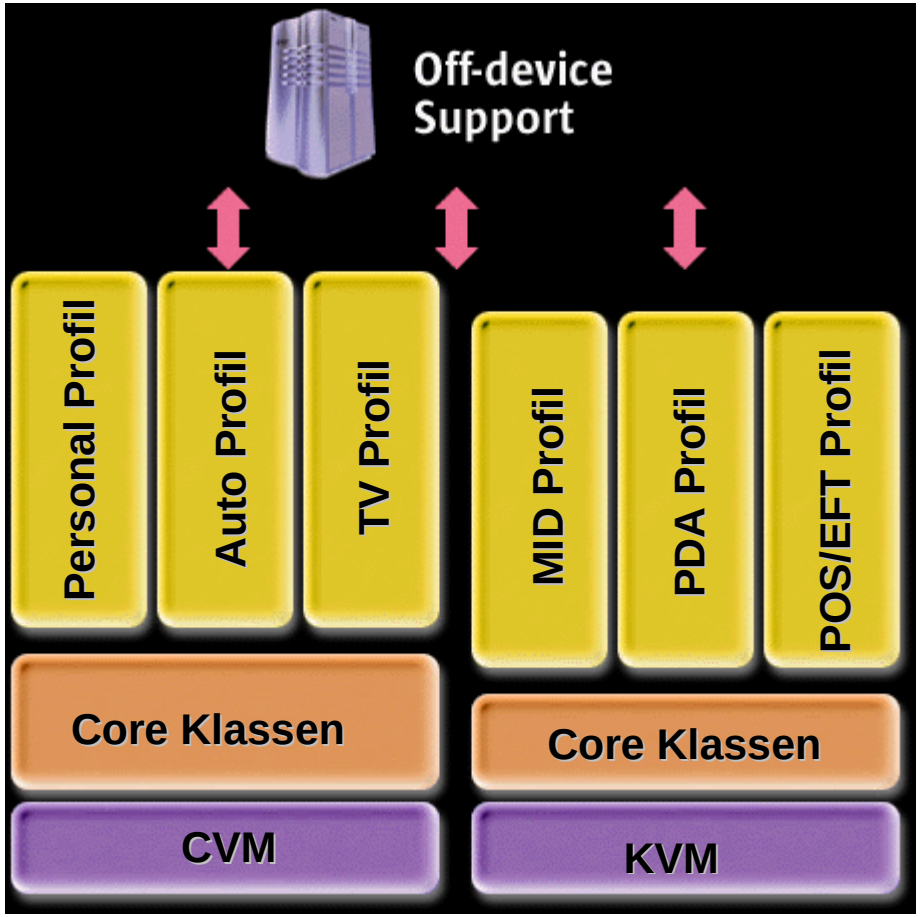
Java Idealvorstellung

- Write Once Run Anywhere
  - Resultat: One Size Fits All
- Praxis:
  - Einheitliche Sprachbasis
  - Spezialisierte Plattformen für Devices
    - Reduzierter Sprachumfang
    - Minimaler Grund-API-Satz
    - Modulare Profile für weitere nötige APIs

Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- J2ME und Webservices

Die J2ME Architektur

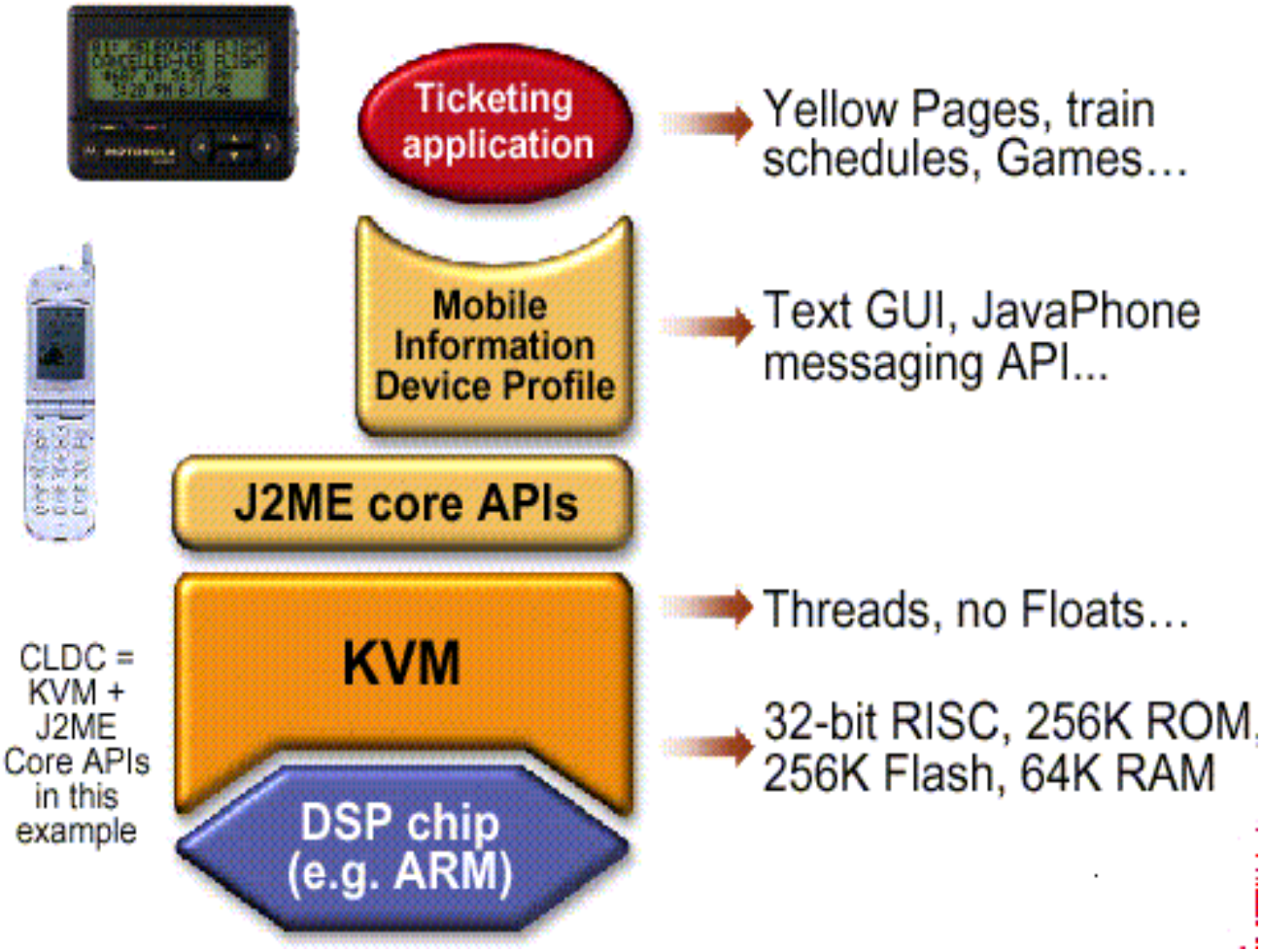


Back-end Module Management und Distribution

Spezifische Industrie- und AnwendungsAPIs *Profile:* z.B. TV, Handy, PDA…

J2ME basiert auf Grund-*Konfigurationen*

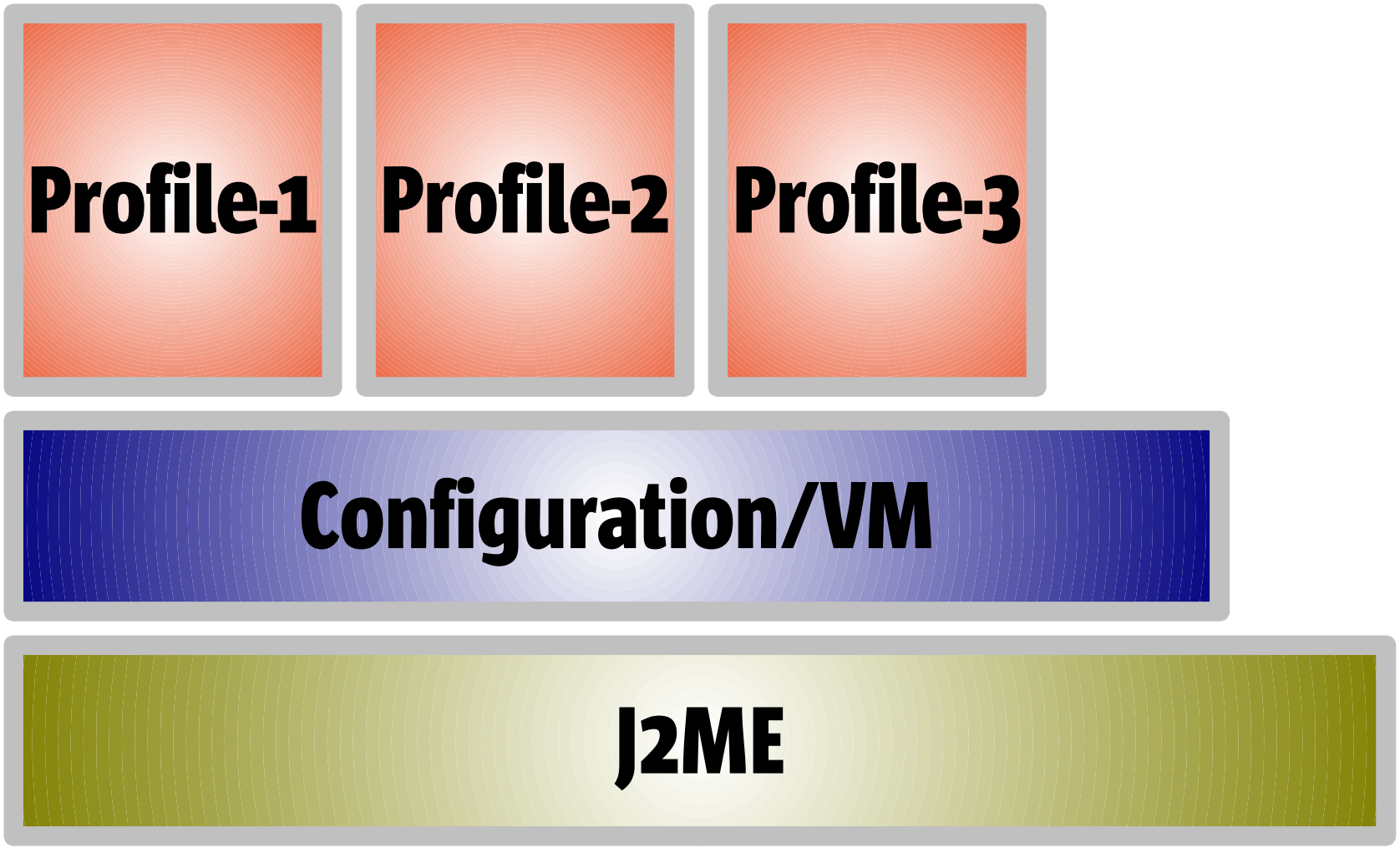
Die Architektur - Beispiel MIDP



Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- J2ME und Webservices

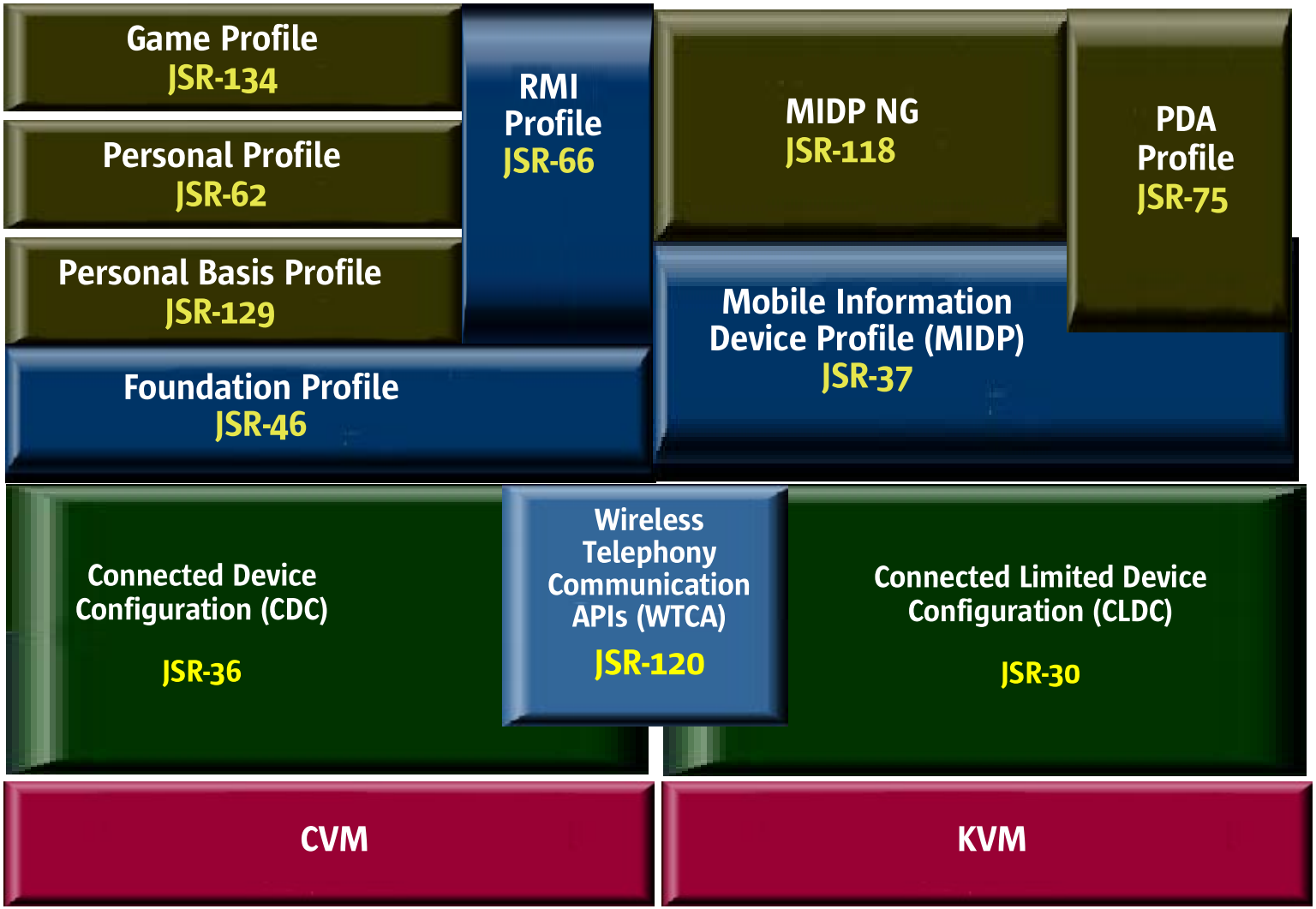
J2ME Konfigurationen & Profile



Wozu sind Profile da?

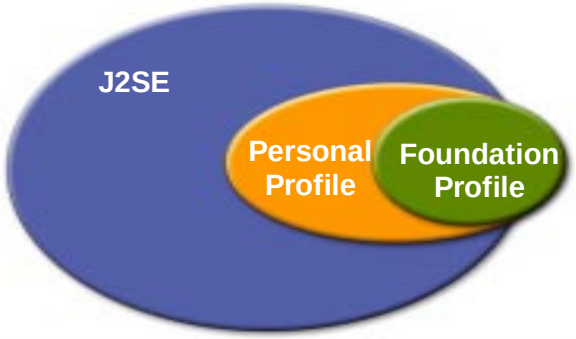
- Stellen APIs zur Verfügung
- Berücksichtigung der Eigenschaften von speziellen Devices, z.B.
  - Handies, Pager,PDAs
  - POS-Terminals
  - Webphones
  - Unterhaltung- und Nav.-Systeme im Auto
  - Telematik-Systeme
- Aufbauend auf Konfigurationen
- Abgeleitet von entsprechenden J2SE Funktionen

J2ME Profile Stack



Foundation Profile

- Definiert grundlegende Java Funktionalitäten
  - nahezu J2SE
  - aber kein GUI
- Komplettiert die CDC



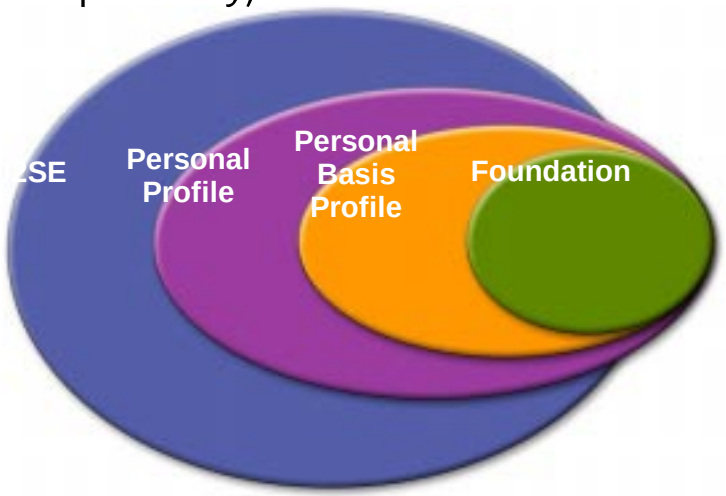
  - java.lang Wesentliches aus vollem java.lang
  - java.util ZIP-Support, Timer
  - java.net TCP-Sockets und HTTP
  - java.io Wesentliches aus vollem java.io
  - java.text Volles I18N
  - java.security Code Signing und Certificates

Personal Profile

- Setzt auf dem Foundation Profile auf
- Design-Ziele
  - Aufwärtskompatibel zu Personal Java
  - GUI Features
    - AWT
    - Beans
    - Applets aus Personal Java

Personal Basis Profile

- Der kleine Bruder des Personal Profile
- Abgrenzung zu Personal Profile
  - AWT
    - keine heavy-weight Widgets (nur Layout und Graphics)
    - ausreichend für
      - HAVi (Home Audio / Video Interoperability)
      - JFC/Swing
- Appl.-Modelle
  - Main, Xlet
  - keine Applets



Java2 Devices

SONY
SO503i

N C
N503i

LG Telecom
iBook

Compaq
iPaq

RIM
Blackberry

Nokia
Communicator 9210

Siemens
U45K

Motorola
iDen i85s

Motorola
Accompli 009

Palm

handspring
Visor

Mitsubishi
D503i

Fujitsu
F503i

Panasonic
P503i

Inventec
ip88

SONY
Clie

Sharp
Zaurus

Motorola
Accompli 008

T

Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- J2ME und Webservices

CLDC als Zielplatform

Designgrundlagen für die CLDC

- 16 bit m-Prozessor
- 160 kB - 512 kB ROM / Flash
- 32 kB – 256 kB RAM
- Typisch Batteriebetrieb
- Netzwerkverbindung
  ggf. nicht immer verfügbar !!!
  (Funk, RS232, Bluetooth mit typischerweise 9600 Baud GSM)
- Minimales Display

Das generische Connection Modell

- Definiert in dem Packet javax.microedition.*
- Enthält ein generisches Rahmenmodell für „low level“ Ein-/Ausgabe und Kommunikation (Datei, Seriell, Datagram, Socket, HTTP)
- Spezifikation des Protokols über „Uniform Resource Indicators“ URIs nach RFC2396

```
Connector.open("<protocol>:<address>;<parameters>");

http://www.sun.com
file:/foo.dat
comm:0;baudrate=9600
```

Das generische Connection Model

- Keine Implementierung
  (Implementierungen einzelner Protokolle werden in den Profilen festgelegt)
- Erzeugung von „Connections“ über statische „Factory“ Methoden in der Klasse Connector

```
Connection open(String name);
Connection open(String name, int mode);
Connection open(String name, int mode, boolean
timeouts);
```

Eigenschaften der KVM

- Unterstützung fast aller Java Sprachelemente
  - Byte, boolean, char, short, int, long
  - Unicode
  - Strings und StringBuffer
  - Klassen und Schnittstellen
  - Exceptions
  - Threads
  - Garbage Collection GC
  - ByteCode Verifikation und dynamisches Laden von Klassen

Sicherheitsmodell der KVM

Das vereinfachte “Sandbox” Modell der CLDC

- Kein Überladen oder Ersetzen der Systemklassen
  (Geschlossener Satz an Funktionalität
  java.*, javax.microedition.* Namensraum)
- Keine benutzerdefinierbaren Klassenlader
  (Reihenfolge der Klassensuche)
- Kein Nachladen von nicht-Java Methoden
  (Keine öffentliche“native” Schnittstelle)

Einschränkungen der KVM

Unterschiede zur “Classic” VM CVM

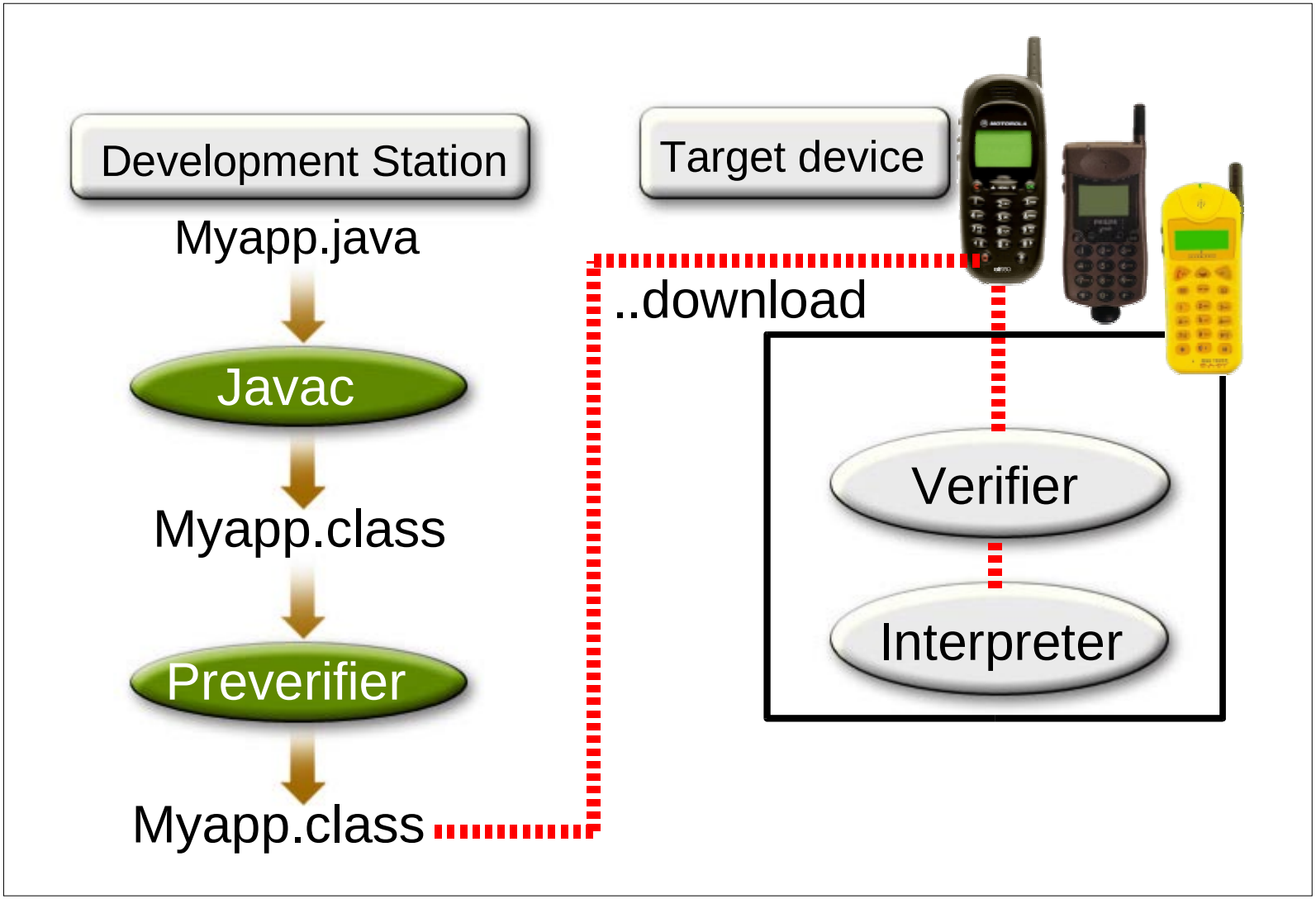
- Keine Gleitkommazahlen
  (kein float, double)
- Keine Finalisierung
  (Object.finalize() nicht vorhanden)
- Keine Reflection
  (Keine Unterstützung für RMI, Objektserialisierung, Profiling JVMPI oder Debugging JVMDI)
- Eingeschränkte Fehlerbehandlung
  (Es weden nur java.lang.VirtualMachineError und java.lang.OutOfMemoryError unterstützt)

Einschränkungen der KVM

Einschränkungen durch das vereinfachten Sicherheitsmodell

- Keine Unterstüzung des JNI
  (“native”-Methoden dürfen nur vom J2ME Hersteller implementiert weden)
- Keine benutzerdefinierten Klassenlader
- Nur einfache Threads
  (Keine ThreadGroup's oder Daemon Threads)
- Keine schwachen Referenzen

Classfile Verification in der KVM



Funktionsweise des Verifizierers

- Zwei Phasen Verifikation
  - Off-Device Pre-Verifikation
  - In-Device Verifikation
- Pre-Verifier
  - Entfernen von jsr / ret Bytecodes durch “inlining” der Subroutinen
  - Einfügen von Stapel-Abbild Attributen in die “.class” Datei (Subattribut des Code- Attributs)
  - Erzeugt eine gültige Klassendatei

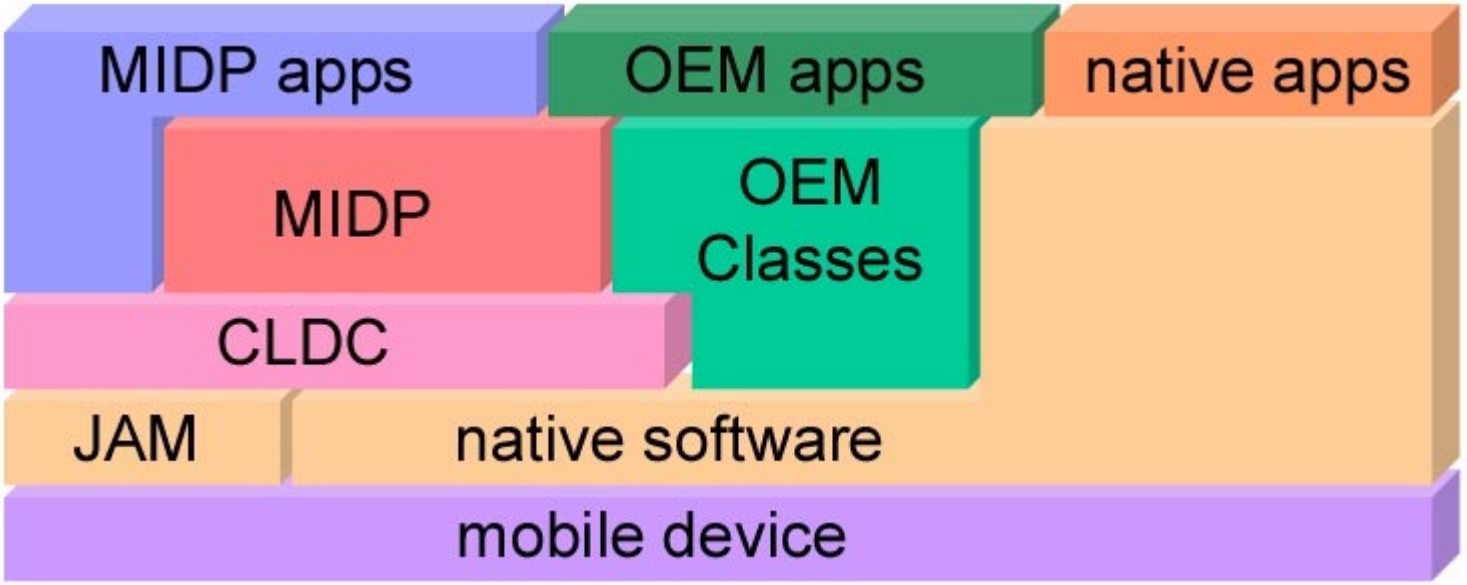
Der Garbage Collector

- In CLDC 1.0
  - 100% vollständig aber nicht 100% exakt
    (Integer auf dem Stapel können als Zeiger interpretiert werden)
  - Non-moving, mark-and-sweep Algorithmus
  - Unterstützt Klassen GC
  - Führt zur Speicherfragmentierung
- Seit CLDC 1.0.2
  - Generationaler GC (voll kompaktifizierend)

Eigenschaften von MIDP

- Erweiterungen der CLDC für Handys, Pager und Kleinst-PDAs
- Drahtlose Verbindung zu einem Netzwerk
- Einfache grafische Benutzeroberfläche
- Nicht flüchtigen Datenspeicher

MID-P Block Diagram



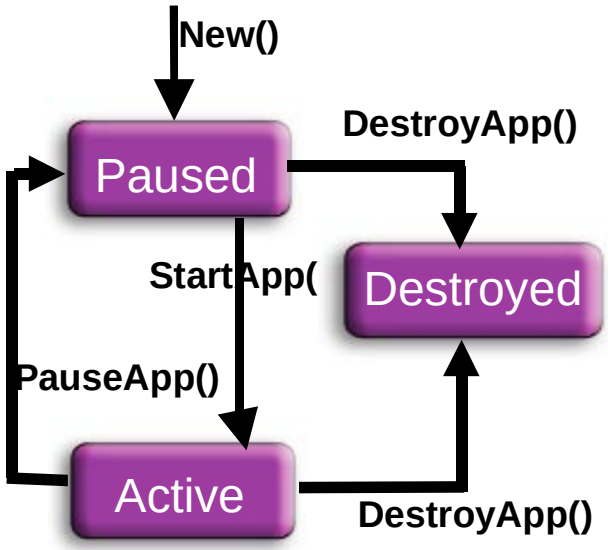
Hardwarevorraussetzungen für MIDP

- Anzeige
  - 96 x 56 Pixel Minimum
  - 1-bit Monochrom oder Farbe
  - 1:1 Seitenverhältnis der Pixel
- Eingabe
  - ITU-T Tastatur, ASCII-Tastatur oder „touch-screen“
- Speicher (zusätzlich zur CLDC)
  - 128 kByte ROM / Flash
  - 32 kByte RAM

MIDlet Lebenszyklus

Application Management Software retrieves MIDlet from a Source

1. Retrieval
2. Installation
3. Launching
   - 3a. MIDlet Instance creation – Paused
   - 3b. MIDlet initialization – Active
   - 3c. MIDlet termination – Destroyed
4. Version Management



Application Management Software installs MIDlet onto device (Security policy verified)

Application Management Software upgrades the MIDlet to a new version, if one is available

Java Applikation Management (JAM)

- Aufgaben:
  - Inspizieren der auf dem Device gespeicherten Applikationen
  - Auswahl und Start der Applikationen
  - Löschen vorhandener Applikationen
- Hochgradig von jeweiligen Device abhängig
- Nicht notwendigerweise Speicherung der Applikationen auf dem Device

MIDP for Palm™OS

- Released am 30. May 2001
- Download von
  http://developer.java.sun.com/developer/earlyAccess/midp_palm
- Zielplattform: PalmOS 3.5
- Beinhaltet folgende Funktionaliät
  - Desktop Utility zum Konvertieren von MIDlets nach PRC
  - MIDP User Preference
  - Samples und Dokumentation

MIDP-Erweiterungen der CLDC

Netzwerk

- MIDP unterstützt Netzwerkverbindung über eine Untermenge von HTTP 1.1
- HTTP wird über TCP/IP oder WAP / i-mode implementiert.
- Datagram Verbindungen können optional über das DatagramConnection Interface implementiert werden

MIDP-Erweiterungen der CLDC

Persistenter Speicher

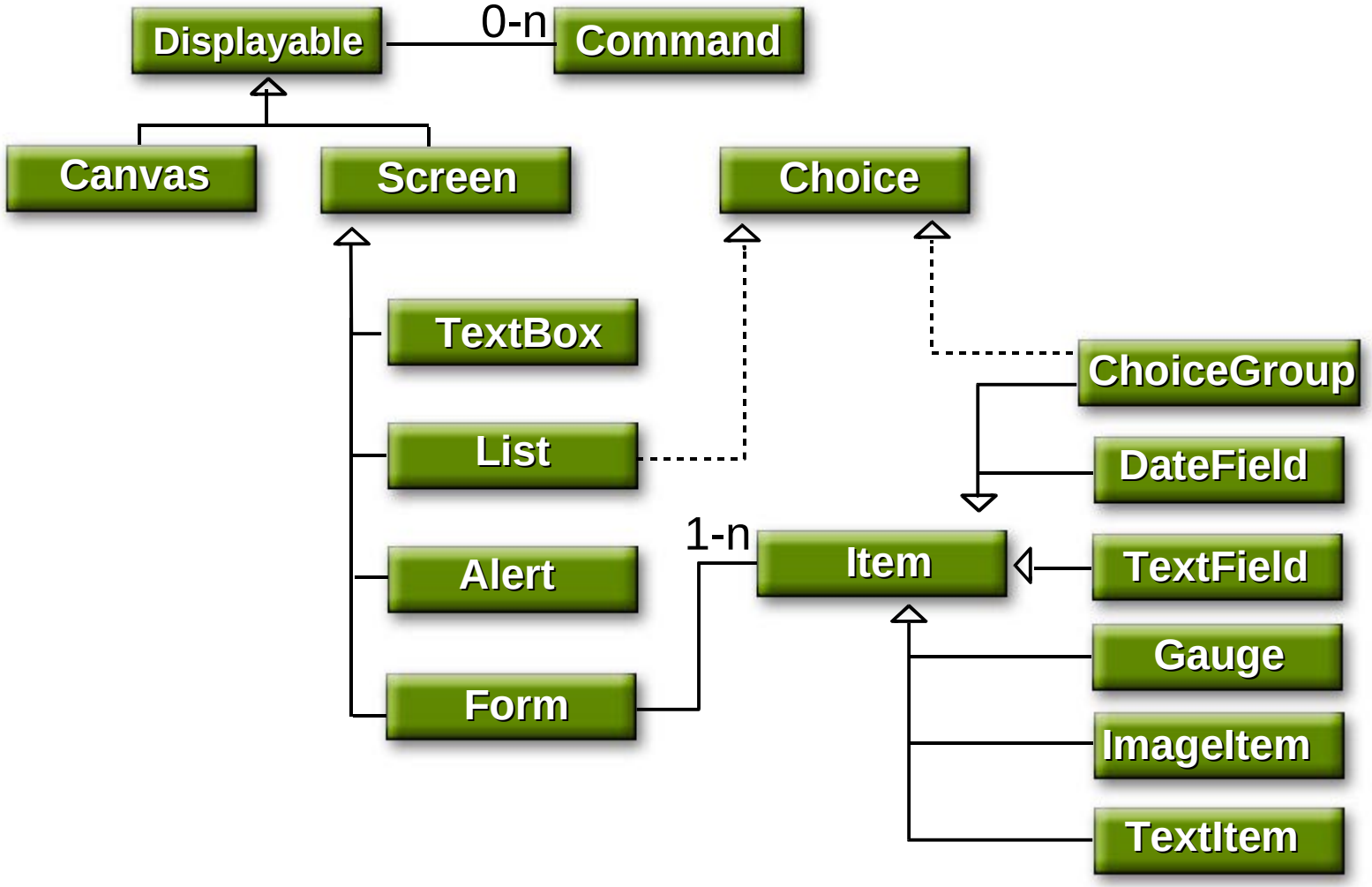
- Persistenter Speicher wird über ein „Record Store“ bereitgestellt.
  Paket javax.microedition.rms
  Es steht folgende Funktionalität zur Verfügung:
  - Speichern/Löschen von Daten
  - Enumeration
  - Listener zur asynchronen Benachrichtigung von Änderungen

UI Design Prinzipien

- Verwendbar in allen Devices
  - Klare Begrenzung auf Devices
  - Kleines Display (einige 10 Pixels x einige 10 Pixel)
  - Nicht alle Devices haben ein Pointing Device
  - Drastisch unterschiedliche Keyboards: Von QWERTY bis herab zu ITU-9 Keypads
- Zielgruppe sind Endanwender (keine Computer-Spezialisten)

MIDP UI Class Hierarchy

javax.microedition.lcdui



Zwei Layers von UI APIs

- High- level API für leichte Portabilität
  - Applikationen sollen auf allen Devices laufen
  - Applikationen sollen auf allen Devices benutzbar sein
  - Kein direkter Zugriff auf Device Features
    - Colors, Screen Sizes, Input Devices
- Low-level API für Applikationen wie Games
  - Drawing Primitives
  - Key Events
  - Developers dürfen Portabilität einschränken um bessere Spielbedingungen zu erreichen

High-level APIs

- High-level APIs arbeiten auf einem hohen Abstraktions-Level aus Portabilitätsgründen
- Zeichenoperationen, Navigation und Scrolling werden von der Implementierung durchgeführt
- Applikationen können nicht auf Input Devices (wie spezielle Tasten) zugreifen

Screen-based Design

- Zentrale Abstraktion für das MIDP's UI ist ein Screen
- Screen kapselt device-spezifisches grafisches Rendering and User Inputs
- Nur ein Screen ist zu einem Zeitpunkt sichtbar
- Implementierung bearbeitet alle Events

3 Typen von Screens

- Generic Screens
  - Class Form
- Screens, die eine UI Komponente kapseln
  - List, TextBox
- Screens used in Low-level APIs
  - Canvas class

Low-level APIs

- Low-level APIs arbeiten, im Gegensatz zu den High-level APIs, auf niedrigem Abstraktions-Level
- APIs sind entworfen für Applikationen, die Kontrolle über Optik, Placement und Funktion von UI-Elementen benötigen
- Ermöglichen Zugriff auf low-level Input Events

Erweiterungen der CLDC

Low-Level API für Spiele

- Volle Kontrolle der Ausgabe auf dem Display
- Zugang zu spezifischen Tasten und Eingabemedien
- Der Entwickler kann zwischen Portabilität und Funktionalität abwiegen
- Es ist möglich die API in einer portablen Art zu verwenden
- Achtung! Umgehen von Abhängigkeiten der Applikation von speziellen Eigenschaften des Geräts

Agenda

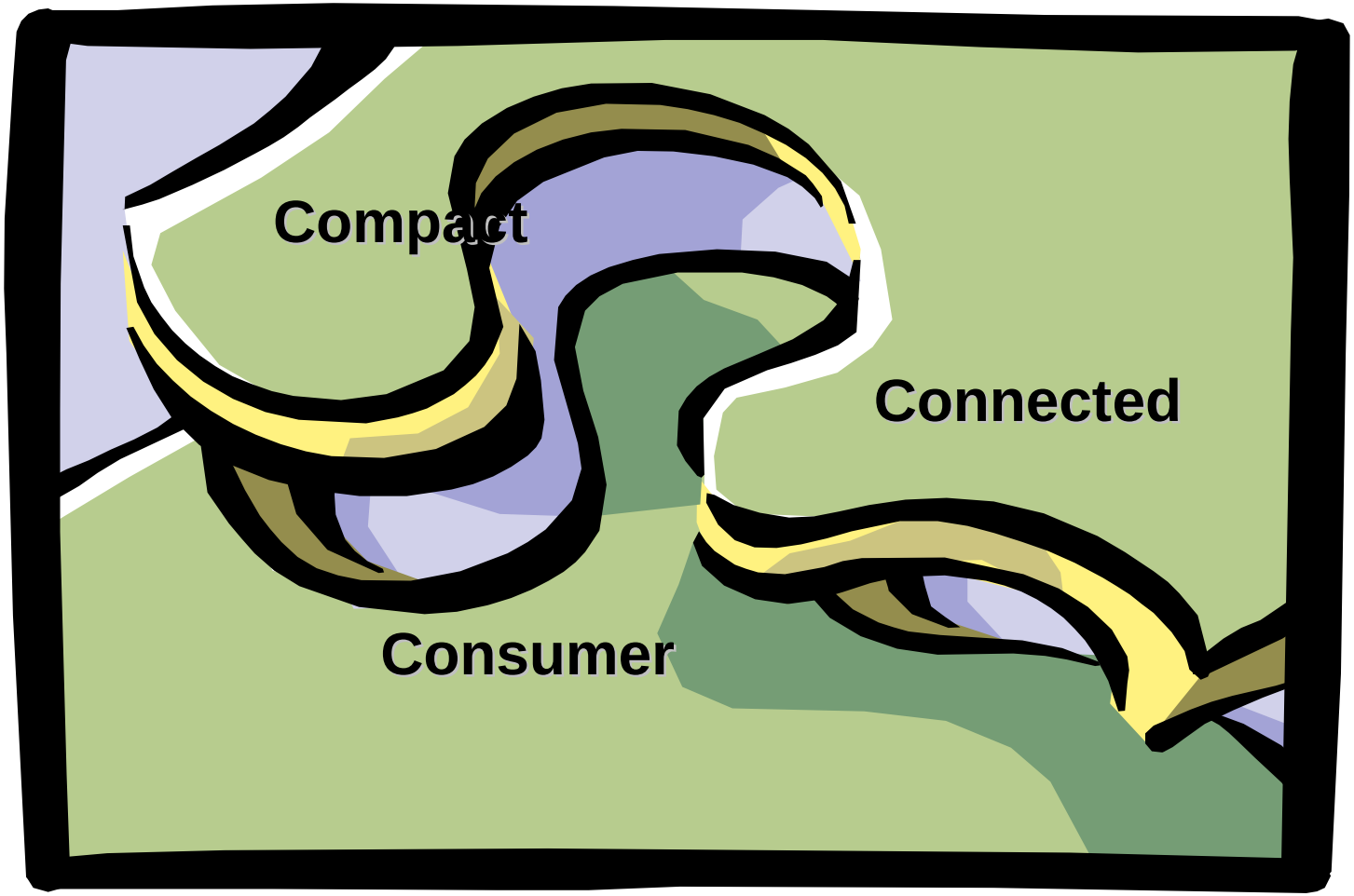
- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- J2ME und Webservices

CDC als Zielplatform

Designgrundlagen für die CLDC

- 16 bit m-Prozessor
- 160 kB - 512 kB ROM / Flash
- 32 kB – 256 kB RAM
- Typisch Batteriebetrieb
- Netzwerkverbindung
  ggf. nicht immer verfügbar !!!
  (Funk, RS232, Bluetooth mit typischerweise 9600 Baud)
- Minimales Display

Was bedeuted das „C“ in der CVM



Die Java-Klassen der CDC

- java.lang VM System Klassen (Object, Thread, etc.)
- java.util darunterliegende Utilities
- java.net UDP Datagram und File URL
- java.io File
- java.text I18n für Java VM Fehler Meldungen
- java.security feinkörnige Sicherheit und Verschlüsselung für Objekt Serialisierung

Zielplatformen der CVM

- Einsatzgebiete:
  - Eingebettete Systeme
  - Settop Boxen
  - Webphones
  - Home Gateways (OSGi)
  - Communicators
- Anforderungen
  - 32 bit CPU (FPU optional)
  - Keine MMU
  - Unterstützung für malloc/threads/synchronisation

Eigenschaften der CVM

- Volle Unterstützung des Java Sprachumfangs.
  - Vollwertige „Blue Book“ VM
  - Definiertes und vorhersagbares Verhalten
  - ~256 kB inklusive Verifier und Classloader
  - Exakt
    - All Pointer sind der VM bekannt.
    - Volle Kompaktifizierung des Heaps.
    - GC kann modular ausgetauscht werden.

Die CVM Klassenbibliothek

- Selbe Quellen wie J2SE mit Anpassungen auf fehlendes AWT
- JVMDI Unterstützung
  - Remote Debugging
    - Verbindung über das JDWP
- Security
- JNI
- Reflection / Serialisierung / RMI
- Schwache Referenzen

Verfügbarkeit

- CLDC/MIDP
  - Solaris, Linux, Windows
  - Palm
- WTK
  - Solaris, Linux, Windows
- CDC
  - Linux
  - VxWorks

Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- **Entwicklungswerkzeuge für J2ME**
- **J2ME und Webservices**

Sun ONE Studio 4, Mobile Edition

First release features

Compact integrated J2ME MIDP/CLDC development environment from Sun Microsystems that is part of a product line that supports Java 2 platform end-to-end

Optimised support for J2ME MIDP/CLDC development

Single download – free

Less resource intensive product – grow as you go

Easy integration for Sun's strategic mobile partners

Product has a roadmap to support mobile applications development

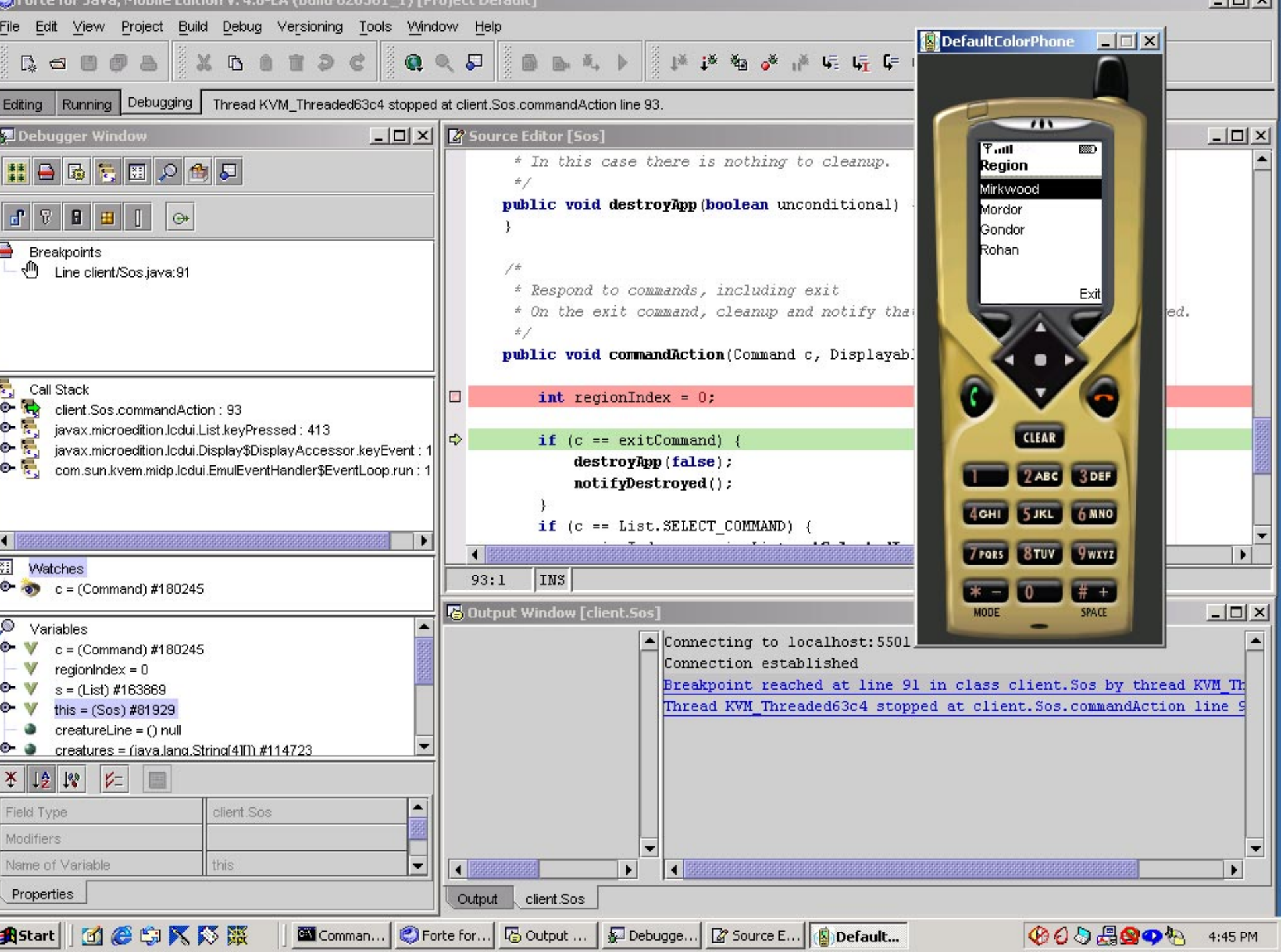
New: Sun ONE Studio 4, Mobile Edition

Support for J2ME MIDP/CLDC development

- Pre-integrated default emulator
  - J2ME compliant emulator based on the CLDC and MIDP Reference Implementations, including debug version of MIDP for Palm OS
  - Device skins with varying form factors
- MIDlet and MIDlet suite templates
- Integrated compilation, preverification and execution of MIDlets and MIDlet suites
- Integrated source-level debugging
- Automatic generation of JAD and .JAR files
- J2ME Code completion

New: Sun ONE Studio 4, Mobile Edition

- Simple target emulator switching – can mount multiple SDKs with Emulator Registry
- Set target emulator heap size constraints
- Team Development support
  - Built-in CVS client, integration to VCS, CVS, VSS
- Apache ANT module – Create ANT build files in XML to automate tasks
- Fully upgradeable to the Sun ONE Studio, Community Edition features through AutoUpdate technology
- Can use Mobile Edition features in Enterprise Edition – End to End development environment from J2ME to J2SE to J2EE
- Industry support – 3rd party emulators from mobile device manufacturers and service operators through a standard interface


File Edit View Project Build Debug Versioning Tools Window Help
Editing Running Debugging Thread KVM_Threaded63c4 stopped at client.Sos.commandAction line 93.
Debugger Window
Breakpoints
Line client/Sos.java:91
Call Stack
client.Sos.commandAction : 93
javax.microedition.lcdui.List.keyPressed : 413
javax.microedition.lcdui.Display$DisplayAccessor.keyEvent : 1
com.sun.kvem.midp.lcdui.EmulEventHandler$EventLoop.run : 1
Watches
c = (Command) #180245
Variables
c = (Command) #180245
regionIndex = 0
s = (List) #163869
this = (Sos) #81929
creatureLine = () null
creatures = (java.lang.String[4][]) #114723
Field Type
client.Sos
Modifiers
Name of Variable
this
Properties
Source Editor [Sos]
* In this case there is nothing to cleanup.
*/
public void destroyApp(boolean unconditional)
}
/*
* Respond to commands, including exit
* On the exit command, cleanup and notify tha
*/
public void commandAction(Command c, Displayab
int regionIndex = 0;
if (c == exitCommand) {
destroyApp(false);
notifyDestroyed();
}
if (c == List.SELECT_COMMAND) {
93:1
INS
Output Window [client.Sos]
Connecting to localhost:5501
Connection established
Breakpoint reached at line 91 in class client.Sos by thread KVM_Th
Thread KVM_Threaded63c4 stopped at client.Sos.commandAction line 9
Output
client.Sos
DefaultColorPhone
Region
Mirkwood
Mordor
Gondor
Rohan
Exit
CLEAR
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
* -
0
+
MODE
SPACE
Start
Comman...
Forte for...
Output ...
Debugge...
Source E...
Default...
4:45 PM

Unterschiedliche Endgeräte

Sun ONE Studio, Mobile Edition Distribution Partners

Emulator / SDK Integrations

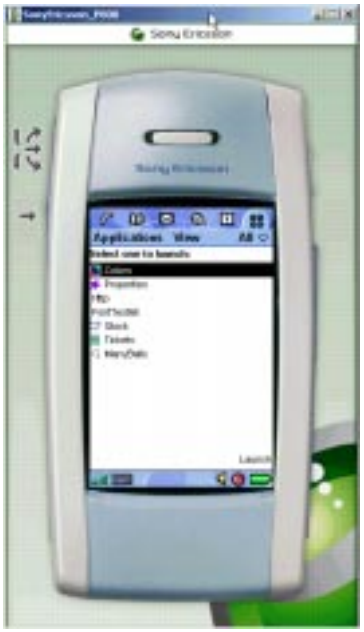
SIEMENS
Mobile

Integration der Siemens Mobility Toolkits

Emulator / SDK Integrations

Unterstützung für Sony Ericsson Java™ Handies

P800, Z700 und T62u über das J2ME Wireless Toolkit



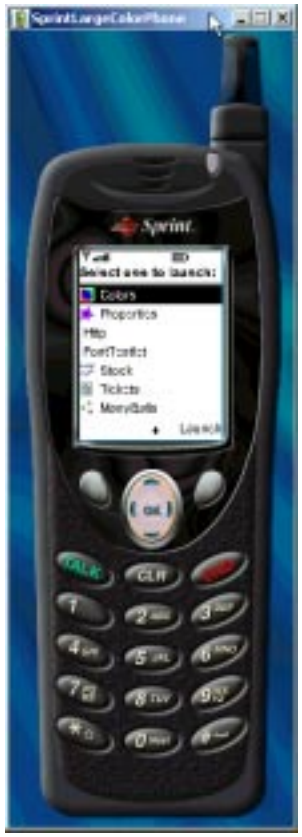
Emulator / SDK Integrations

Unterstützung für Nokia Java™ Handies:

Nokia 6310i und Series 60 SDK Emulation über das J2ME WTK und die Nokia Developer's Suite

Emulator / SDK Integrations

- Unterstützung für Sprint Java™ Handies:
  Sprint PCS Java Wireless Toolkit



Sun ONE Studio 4, Mobile Edition Release Dates

- Early Access (EA): seit 12. März 2002
  http://forte.sun.com/eap/
- Early Access (EA) Japan: seit 25 März 2002
- Announcement : Java One, 27 März 2002
- First Customer Ship (FCS): seit 11.Juni 2002

http://www.sun.com/software/sundev/jde/buy/index.html

Agenda

- Überblick
- J2ME Architektur
  - Konfigurationen und Profile
  - CLDC, MIDP und die KVM
  - CDC und die CVM
- Entwicklungswerkzeuge für J2ME
- **J2ME und Webservices**

SunONE - Sun Open Network

" Standards based software vision, architecture, platform, and expertise for building and deploying Services on Demand"

http://www.sun.com/sunone

Merkmale von Sun ONE

Sun ONE = Services on Demand + Integration + Interoperabilität

- Web-Application-Middleware
- **Webservices** Middleware
- Identity/Context-Services (Liberty Alliance)
- **Web-Client-Modell basierend auf J2ME**
  und Desktop Java Standards
- Weitere Middleware-Interfaces und “Core Web Services”: user/group schema/policy, portal channels, Webtop, communications, e-commerce services, etc.
- System- und Application-management für Webapplikationen und -services
- Plattformservices für Webapplikationen und -services (J2EE-Container)
- Microsoft-, Enterprise-, Legacy-API Interoperabilität
- Entwicklungswerkzeuge zur Entwicklung Java-basierter Webapplikationen und -services, inkl. Einbindung von Legacy-Systemen

http://www.sun.com/sunone

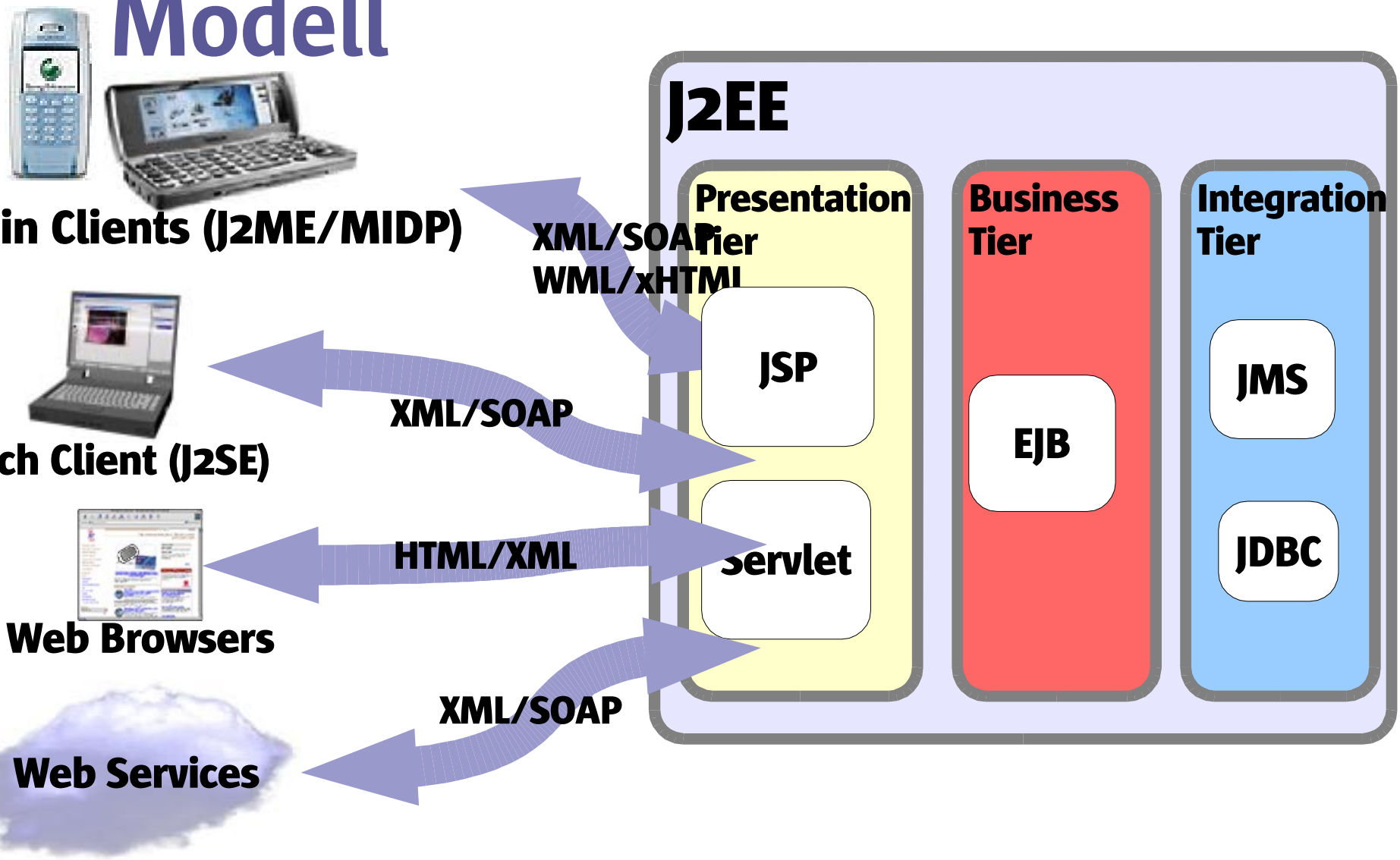
Web Services Charakteristika

- XML-basiert
- Für beliebige Endgeräte zugänglich über das Web (“wireless” oder “wired”)
- Lose gekoppelt (**WSDL**)
- Message-basiert (**SOAP**)
- Auffindbar über eine Registry (**UDDI**)
- Benutzung von Standard-Webprotokollen (HTTP/S)

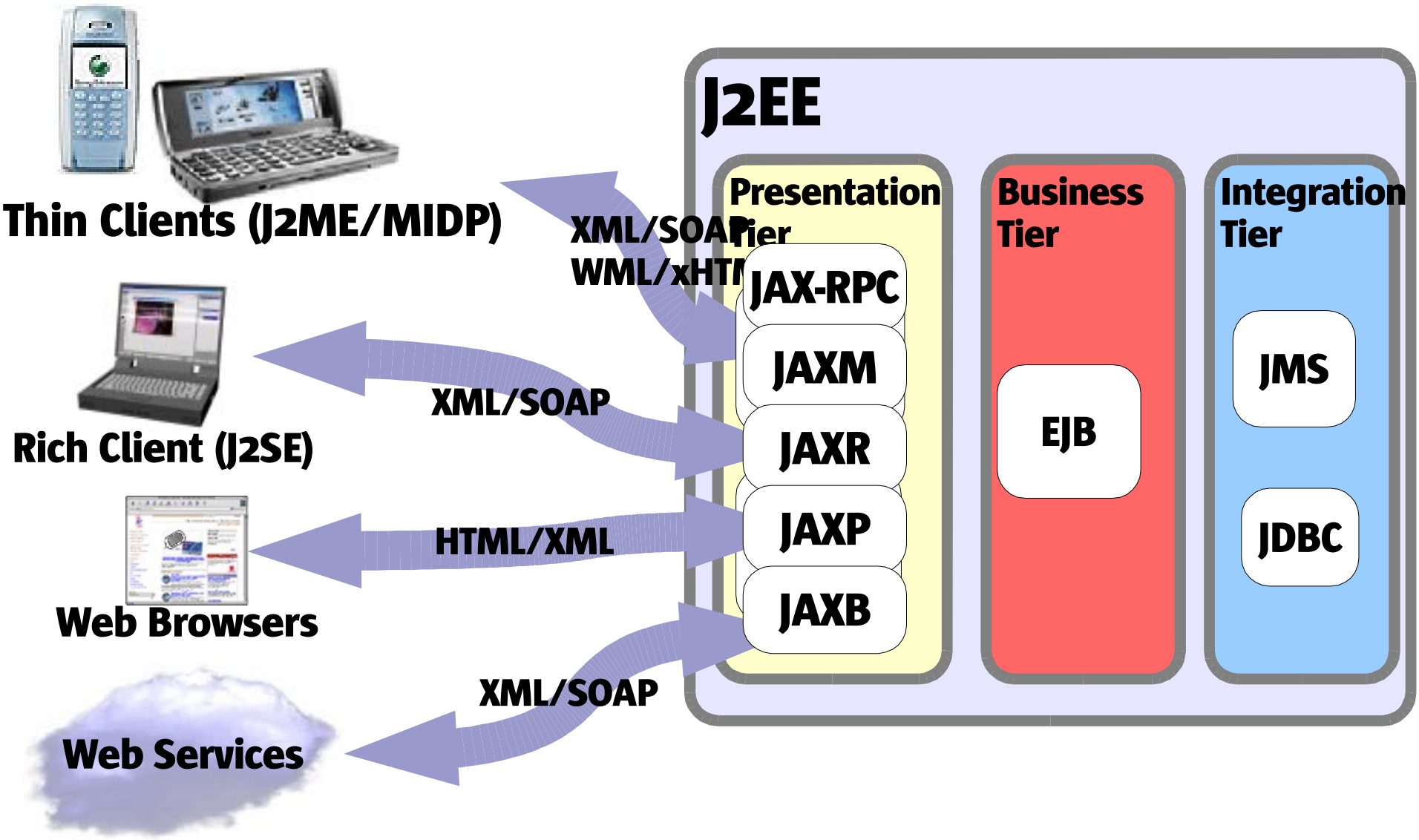
Web Services Standards

- **SOAP** – Simple Object Access Protocol
  RPC in XML formuliert
- **WSDL** – Web Services Description Language
  Standardisiertes XML-Framework, welches die Schnittstellen eines Webservices beschreibt
- **UDDI** – Universal Description Discovery and Integration

Java™ 2 Platform Web Service Modell



The “JAX Pack”


Thin Clients (J2ME/MIDP)
Rich Client (J2SE)
Web Browsers
Web Services
XML/SOAP
WML/xHTML
XML/SOAP
HTML/XML
XML/SOAP
J2EE
Presentation Tier
JAX-RPC
JAXM
JAXR
JAXP
JAXB
Business Tier
EJB
Integration Tier
JMS
JDBC

SOAP – Java™ 2 Approach

- SOAP = RPC in XML
  RPC für Java = RMI
- JAX-RPC unterstützt die Entwicklung von Webservices mittels RMI-SOAP-Mapping
- Kein Paradigmenwechsel, für Java-Entwickler bleibt bzgl. RPC alles beim alten.

JWSDP – Java Web Services Developer Pack

- Das **Java Web Services Developer Pack**

  (Java WSDP bzw. JSWDP) ist ein All-in-one-Download, der die Kerntechnologien für die Entwicklung Java-basierter Webservices enthält.

http://developer.java.sun.com

JWSDP - Inhalt

- JavaServer Pages™ Standard Tag Library (JSTL)
- **Java XML Pack**, Java API for XML Messaging (JAXM),Java API for XML Processing (JAXP) ,Java API for XML Registries (JAXR) ,Java API for XML-based RPC (**JAX-RPC**)
- Tomcat (Java Servlet- und JavaServer Pages™ Web-Container samt Tools)
- ANT (Build-Management Werkzeug)
- Deploytool (Deploymentwerkzeug für Webapplikationen)
- UDDI-Registry Server

SOAP – CLDC/MIDP Optionen

- JSR-172 – J2ME Web Services Specification
- MIDP bietet per se keine Unterstützung für XML/SOAP
- Die meisten SOAP-Implementierungen wurden für J2SE entworfen
- Eine Reihe von XML-Projekten beschäftigt sich mit CLDC/MIDP als Zielumgebung

J2ME CLDC/MIDP XML Projects

Project	MIDP	Type	URL
kXML/kSOAP 1.2	Yes	Pull	http://www.kxml.org, http://www.ksoap.org
kXML 2.0 alpha	Yes	Pull	http://www.kxml.org
ASXMLP 020308	Yes	Push, Model	http://www.alsutton.com/software/xmlparser
Xparse-J 1.1	Yes	Model	http://www.webreference.com/XML/tools/xparse-j.html
NanoXML 1.6.4	Patch	Model	http://nanoxml.sourceforge.net
TinyXML 0.7	No	Model	http://www.gibaradunn.srac.org/tiny
MinML 1.7	No	Model	http://www.wilson.co.uk/xml/minml.htm

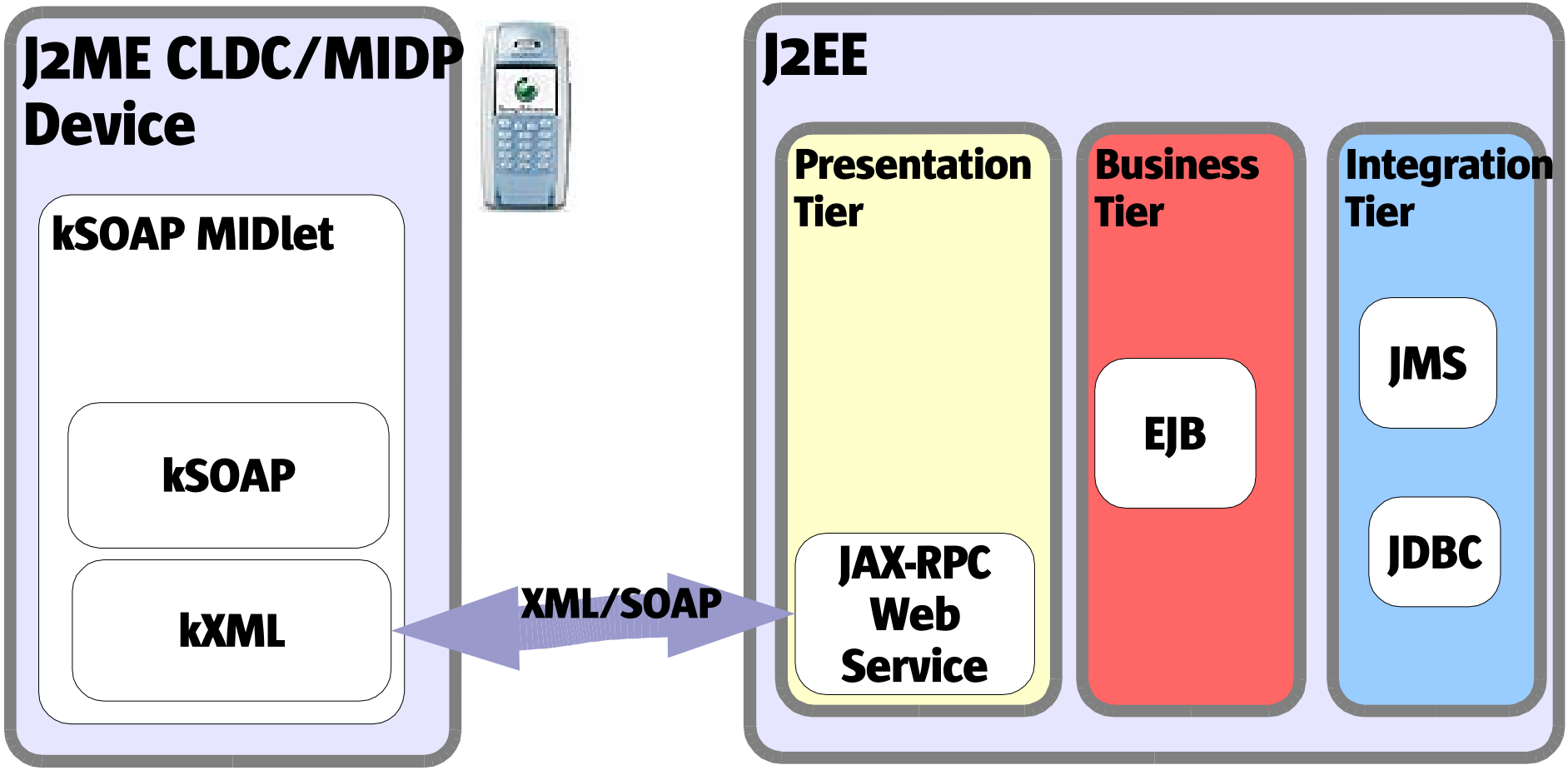
http://**wireless.java.sun.com**/midp/articles/parsingxml

kSOAP/kXML

- Open Source
- SOAP-Implementierung für J2ME
- kSOAP ist ca. 19 kB klein
- setzt kXML voraus (mind. 21 kB)
- Low-level RPC Implementierung (keine WSDL-Unterstützung)

http://www.ksoap.org

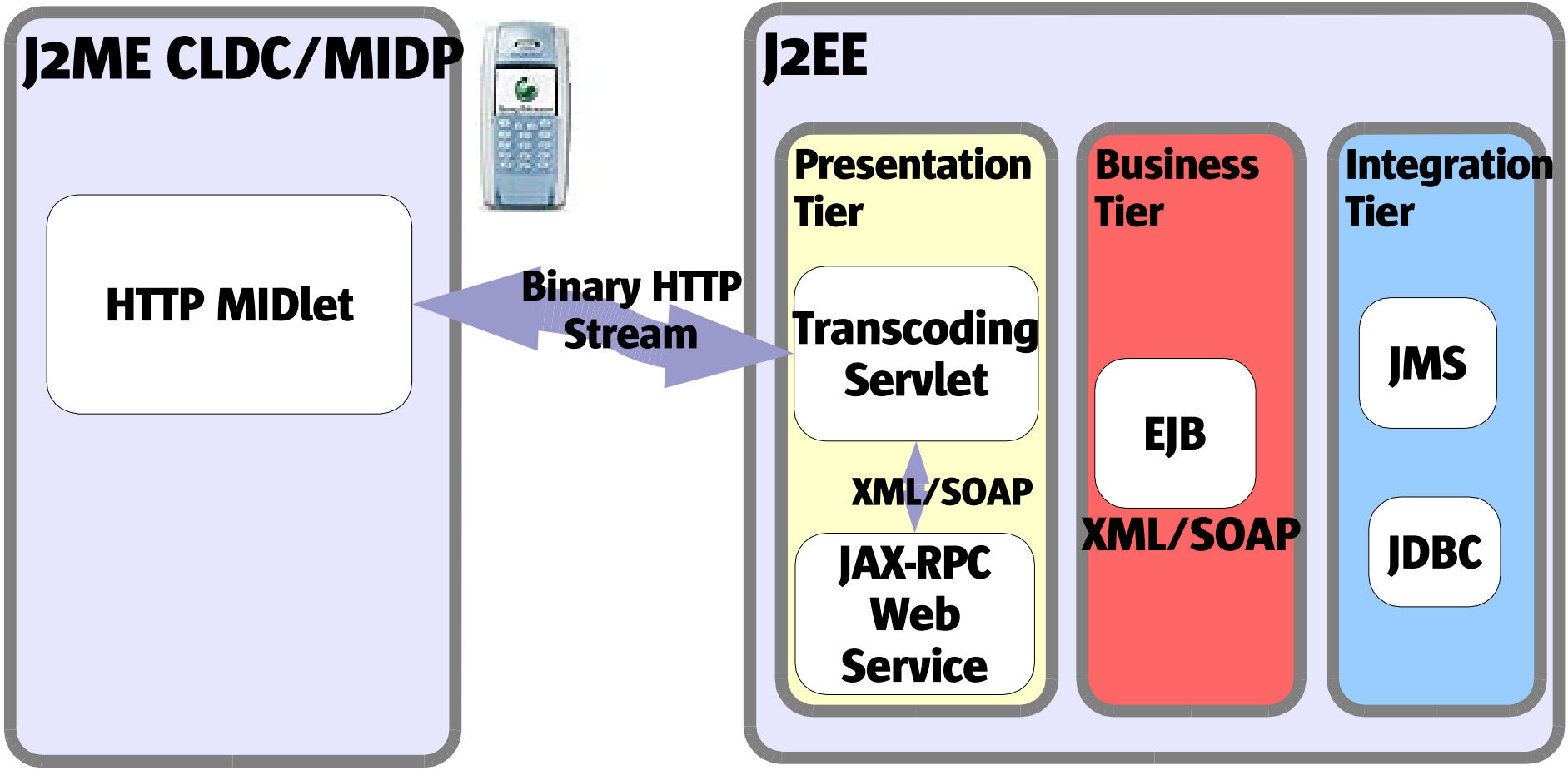
kSOAP


J2ME CLDC/MIDP Device
kSOAP MIDlet
kSOAP
kXML
XML/SOAP
J2EE
Presentation Tier
JAX-RPC Web Service
Business Tier
EJB
Integration Tier
JMS
JDBC

kSOAP Performance (Nokia-9210)

	MIDlet size	Request Setup	Request	Request Size	Reply Size
SOAP	~ 48 kB	~ 12.0 sek	~ 3.8 sek	529 bytes	594 bytes
non-SOAP	~ 5 kB	~ 4.7 sek	~ 2.1 sek	18 bytes	34 bytes
	960%	255%	181%	2939%	1747%

Lösung: Binary Transcoding



Weshalb Transcoding ?

- Dynamik von WSDL nicht überall notwendig
- Limitierender Faktor: GRÖSSE in KByte
- Throughput ist entscheidend
- Komplexität steigt ggü. kSOAP nur unwesentlich

Useful Links

- SunONE –
  http://www.sun.com/sunone
- Sun One Architecture Guide -
  http://wwws.sun.com/software/sunone/docs/arch/index.html
- JWSDP –
  http://developer.java.sun.com
  http://java.sun.com/webservices/webservicespack.html
- Wireless Java Development -
  http://wireless.java.sun.com
- kSOAP & kXML -
  http://www.ksoap.org, http://www.kxml.org
- J2ME Allgemein
  http://www.billday.com/j2me/index.html

Christian Eisoldt

christian.eisoldt@sun.com

Sun microsystems

We make the net work.