

SCHRIFTLICHE AUSARBEITUNG DES VORTRAGES FÜR DAS PROSEMINAR „MOBILE JAVA“

CLDC vs. CDC

Eine Gegenüberstellung der beiden Configurations der Java 2 Micro Edition

von Fabian Reiß

BETREUUNG: MARTIN GUMHOLD, DR. FRANK KARGL

Inhaltsverzeichnis

1. Einführung	2
1.1 Einsatzbereich der Java 2 ME	2
1.2 Aufbau der Java 2 ME	2
1.2.1 Die Java Virtual Machine.....	3
1.2.2 Die Configuration-Ebene	3
1.2.3 Die Profile-Ebene	4
2. Die CLDC im Detail	5
2.1 Die Spezifikation der CLDC	5
2.2 Aufgaben der CLDC	5
2.3 Klassen der CLDC	6
2.3.1 CLDC-spezifische Klassen.....	6
2.4 Das Sicherheitssystem der CLDC	7
2.4.1 Das Sicherheitssystem auf unterster Ebene	8
2.4.2 Das Sicherheitssystem auf der Ebene der Anwendungen.....	9
2.5 Weitere Einschränkungen der CLDC.....	9
2.5.1 Fließkommazahlen	9
2.5.2 finalize-Methode	9
2.5.3 Fehler und Exceptions.....	9
2.5.4 Class-Loader	10
3. Die CDC im Detail	10
3.1 Zielgruppe der CDC.....	10
3.2 Einschränkungen der CDC	10
3.3 Aufbau der CDC	10
3.4 Das Sicherheitssystem der CDC	11
3.4.1 Virtual Machine Security.....	11
3.4.2 Signed Classes.....	11
3.4.3 Policy-based security.....	11
3.4.4 Cryptography	11
3.4.5 Certificate management	11
4. Die Configurations im Überblick.....	11
4.1 Einschränkungen bei der CLDC im Vergleich zum J2SE-Standard	11
4.2 Einschränkungen bei der CDC im Vergleich zum J2SE-Standard	11
4.3 Tabellarische Gegenüberstellung	12
5. Abbildungsverzeichnis	13
6. Literaturverzeichnis	13

1. Einführung

1.1 Einsatzbereich der Java 2 ME

Tragbare Geräte nehmen in unserem heutigen Leben einen wichtigen Platz ein. Nicht nur beruflich, auch privat sind wir immer häufiger auf Mobiltelefone, Organizer (PDAs) oder auch Pager angewiesen. Die Zahl dieser kleinen Informationsgeräte wächst ständig, im Jahre 2003 wurde allein die Gruppe der Handynutzer auf 1 Milliarde weltweit geschätzt. Das besondere am Markt der mobilen Endgeräte ist nicht nur sein schnelles Wachstum, er zeichnet sich auch durch ein breites Spektrum der Gerätearten aus. Dazu kommt, dass der technische Fortschritt in diesem Bereich rapide voranschreitet und die Geräte innerhalb kürzester Zeit immer kleiner und vor allem leistungsfähiger werden. Synchron zum Leistungszuwachs der Geräte wird auch deren Betriebssoftware immer aufwändiger. Im Gegensatz zu früher, als man Handys mit fest verdrahteter Software herstellte, bietet sich heute die Möglichkeit, mobile Anwendungen individuell anzupassen. Leistungsstarke Anwendungen benötigen allerdings eine entsprechende starke Laufzeitumgebung für den mobilen Betrieb und eine möglichst einfache Entwicklungsplattform für die Programmierung.

Java 2 Micro Edition (J2ME) von SUN bietet eine entsprechend potente Softwareplattform zur Erstellung personalisierter, gerätespezifischer und leistungsbezogene Anwendungen für **embedded devices** bzw. **vernetzte Informationsgeräte**. Java 2 ME basiert hierbei auf der einige Jahre zuvor entwickelten **Java 2 Standard Edition (J2SE)**. J2ME stellt jedoch eine Einschränkung der J2SE dar, da sie speziell für Geräte mit limitierten Ressourcen ausgerichtet ist, beispielsweise in Bezug auf die Speicherausstattung oder die Prozessorleistung.

1.2 Aufbau der Java 2 ME

Das Einsatzgebiet der Java 2 Micro Edition ist ziemlich vielfältig und wird geprägt von den Hardwarevoraussetzungen der einzelnen Geräte. Hierbei stehen drei Geräteeigenschaften im Vordergrund, anhand deren sich eine Differenzierung durchführen lässt:

- Systemarchitektur
- Leistungsfähigkeit
- Anwendungsfeld

Der strukturelle Aufbau der Java 2 ME ist hinsichtlich der unterschiedlichen Anforderungen im Bereich der vernetzten Endgeräte entsprechend flexibel gestaltet. Die Plattform der J2ME besteht aus drei aufeinander aufbauenden modularen Ebenen, die je nach Anforderung entsprechend angepasst werden können. Abbildung 1.1 zeigt einen Überblick dieser Struktur mit einer exemplarischen Anpassung in Klammern.

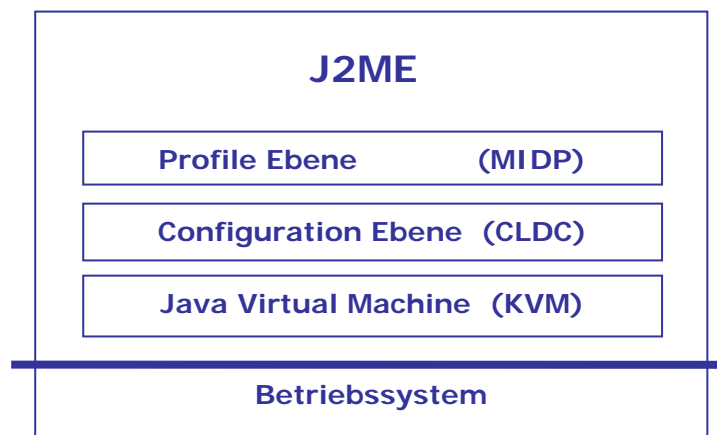


Abbildung 1: Struktur der Java-ME-Plattform

1.2.1 Die Java Virtual Machine.

Wie bei allen Java-Plattformen ist die **Java Virtual Machine** die zentrale Komponente zur Ausführung einer Java-Anwendung. Der Java-Compiler (javac) erzeugt kein Programm in ausführbarer Maschinsprache, sondern bereitet lediglich Byte-Code für den Interpreter vor. Die Virtual Machine stellt dabei sicher, dass alle Java-Funktionen in korrekter Weise auf dem System zur Ausführung gelangen. Bei der J2ME ist die Virtual Machine an das Betriebssystem angepasst und unterstützt eine spezielle Configuration der darüber liegenden Schicht. Für jede Configuration gibt es somit eine eigene Virtual Machine, wobei beide in ihrer Gestaltung eng miteinander verknüpft sind, das heißt die Configuration bestimmt den Umfang der dazugehörigen Virtual Machine. Zur Zeit existieren in der Java Micro Edition zwei unterschiedliche Virtual Machines:

- Die **CLDC HotSpot™ Implementation**¹⁾ ist im Konzept der J2ME ein sehr wichtiges Element, da sie sehr kompakt ist und speziell für Geräte mit limitierten Ressourcen entwickelt wurde. Die CLDC HotSpot™ Implementation basiert auf der zuvor entwickelten **KVM**, besitzt allerdings eine bessere Performanz bei gleichem Speicherbedarf. Das Hauptziel bei der Entwicklung der KVM war die Erschaffung einer kleinstmöglichen "vollständigen" Virtual Machine, die alle Kernpunkte von Java beinhaltet und dennoch maximal 40-80 KB Speicher beansprucht. Zu der Zielgruppe der CLDC HotSpot™ Implementation zählen Handys, Pager und Organizer, die allesamt 16/32-bit Prozessoren und nur wenige hundert Kilobyte Speicher besitzen. Durch die Bemessung des Speichers in Kilobyte leitet sich der Name der KVM ab: Kilobyte Virtual Machine.
- Die **CDC HotSpot™ Implementation**¹⁾ ist ein verbesserter Nachfolger der zuvor entwickelten **CVM** und stellt eine voll leistungsfähige Java Virtual Machine dar, wie sie bei J2SE zum Einsatz kommt. Im Kernpunkt der Entwicklung der CDC stand nicht eine kleinstmögliche Java-Plattform zu entwickeln, sondern möglichst kompatibel zur J2SE zu sein. Allerdings wurde auch die CDC HotSpot™ Implementation für Geräte mit limitierten Ressourcen optimiert, insbesondere in Sachen Performanz und Speicherbedarf. Zu der Zielgruppe der CDC HotSpot™ Implementation zählen high-end PDAs, Bildtelefone und Auto-Navigations-Systeme, die allesamt 32-bit Prozessoren und mehr als 2 MB Speicher besitzen.

1.2.2 Die Configuration-Ebene

Wie bereits erwähnt, existiert für die Anwendungen der Java 2 Micro Edition eine große Anzahl unterschiedlicher Endgeräte, die vielfältige Anforderungen an den strukturellen Aufbau der J2ME stellen. Bei diesem breiten Gerätespektrum ist es sinnvoll, gleichartige Geräte zusammenzufassen und jeder Gruppe eine speziell angepasste Version der Java 2 ME zur Verfügung zu stellen. Diese Gerätegruppen werden **Configurations** genannt. Die „horizontale“ Unterteilung der Endgeräte findet hier auf Basis von Hardwaremerkmalen statt. Dabei definiert jede Configuration eine minimale Anforderung an Merkmalen, die ein dazugehöriges Gerät erfüllen muss, beispielsweise in Bezug auf die Größe des Hauptspeichers oder die Leistung des Prozessors. Anhand dieser Basis an unterstützten Hardwaremerkmalen werden von der Configuration folgende Angaben festgelegt:

- Die Merkmale der Java-Programmiersprache
- Die Merkmale der Java-Virtual-Machine
- Die unterstützten Java-**APIs (Application Programming Interfaces)**.

Dies macht deutlich, dass eine Configuration ein ziemlich komplexer Teil der Java 2 Micro Edition darstellt. Schon eine kleine Änderung in der Spezifikation einer Configuration kann aufwändige Anpassungen im internen Design der dazugehörigen Virtual Machine nach sich ziehen.

Zum aktuellen Zeitpunkt stehen für die Entwicklung zwei unterschiedliche Configurations zur Verfügung, die jeweils eine bestimmte Gerätegruppe abdecken:

¹⁾ Die Begriffe CDC und CLDC werden im nächsten Abschnitt genauer definiert und erläutert.

- Die **CLDC (Connected Limited Device Configuration)** deckt den Markt der vernetzten, mobilen Informationsgeräte ab. Dazu gehören beispielsweise Mobiltelefone, Organizer und Pager. Diese Endgeräte haben im Vergleich zu Desktop PCs eine sehr einfache Benutzerschnittstelle, eine sehr geringe Speicherausstattung, eine geringe Bandbreite und oftmals instabile Netzwerkverbindungen – weswegen diese Geräte nicht auf dem TCP/IP-Standard basieren. Eine weitere Besonderheit dieser Geräte ist, dass sich die Gerätearten von Generation zu Generation immer ähnlicher werden – insbesondere Handys und PDAs besitzen immer öfter die gleichen Charakteristika. Eine Folge dieser fortschreitenden technologischen Konvergenz der Computer- und Telekommunikationsindustrie wird sein, dass man zukünftige Endgeräte dieser Kategorien nicht mehr anhand der eigentlichen Funktionalität unterteilt, sondern nur die Leistungsspezifikationen wie zum Beispiel die Speicherausstattung oder die Prozessorleistung in Betracht zieht.
- Zu der **CDC (Connected Device Configuration)** gehören alle Informationsgeräte, die nicht in der CLDC erfasst sind. Das sind Geräte, die ebenfalls auf die eine oder andere Weise vernetzt sind und Informationen mit anderen Quellen austauschen, allerdings sind sie nicht mobil und besitzen nicht die CLDC-üblichen Einschränkungen. Zur CDC gehören beispielsweise Fernseh-Digitalempfänger, Bildtelefone mit Internetanschluss und Autonavigationssysteme. Diese Geräte haben im Vergleich zu den mobilen Geräten der CLDC eine größere Speicherausstattung, dauerhafte Netzwerkverbindungen und vielfältige Benutzerschnittstellen. Aufgrund der dauerhaften und oftmals sehr leistungsstarken Netzwerkverbindung dieser Geräte, basieren die Netzwerkprotokolle vermehrt auf TCP/IP. Um eine Aufwärtskompatibilität zwischen den beiden Configurations zu gewährleisten, wurde die CDC als Obermenge der CLDC deklariert.

Wie bereits erwähnt, existieren zur Zeit nur die beiden beschriebenen Configurations. Sie wurden von einer Expertengruppe innerhalb des **JCP-Programms (Java Community Process)**, im Jahre 2000 definiert, zu der 18 namhafte Hersteller von Mobiltelefonen beziehungsweise Organizern gehörten. Das JCP ist eine offene Standardisierungsorganisation und überwacht die Entwicklung der Java 2-Technologie. Falls im Laufe der Zeit neue Geräte entstehen, die sich in keine der bisher definierten Kategorien gruppieren lassen, dann besteht die Möglichkeit eine neue, optimierte Virtual Machine und Configuration in J2ME zu definieren. Dieser Standardisierungsprozess wird ebenfalls innerhalb des JCP ablaufen.

1.2.3 Die Profile-Ebene

Waren die bisher angesprochenen Schichten entscheidend für die Ausführung des Programmcodes und die Einteilung der Geräte in Gruppen, so ist die Ebene der **Profile** die wichtigste Schicht für den Benutzer und den Entwickler von Anwendungen. Ein Profil setzt auf einer Configuration auf und legt für eine spezielle Geräteart bestimmte Mindestanforderungen der Java 2 Micro Edition fest. Beispielsweise existiert in der CLDC ein Profil für Handys, das aufführt welche APIs bei allen Handys zur Verfügung stehen müssen. Man kann sagen, Profile dienen zur „vertikalen“ Unterteilung der einzelnen Marktsegmente. Der Hersteller kann im Laufe der Entwicklung eines Gerätes entscheiden, welche Profile von dem Gerät unterstützt werden sollen, muss dann aber auch alle dazugehörigen Merkmale umsetzen. Der Vorteil für den Benutzer ist, dass jede Anwendung die für ein bestimmtes Profil geschrieben wurde, auf allen Geräten eingesetzt werden kann, die dieses Profil unterstützen. Beispiele einiger wichtiger Profile:

- MIDP = Mobile Information Device Profile (Erweiterung der CLDC, für Handys optimiert)
- Foundation Profile (Erweiterung der CDC, Basis für weitere Profile der CDC)
- PDA Profile (Erweiterung der CLDC, für PDAs optimiert)

2. Die CLDC im Detail

Die Aufgaben der Connected Limited Device Configuration nehmen im Gesamtkonzept der Java 2 Micro Edition eine zentrale Rolle ein. Einer der wichtigsten Gründe für die Entwicklung der J2ME ist die Festlegung des Standards einer Java-Plattform für kleine, vernetzte Geräte mit eingeschränkten Ressourcen. Dies ist die Aufgabe der CLDC, die damit die Anforderung erfüllt, dynamische Java-Anwendungen auf Geräten wie zum Beispiel Handys oder Organizer auszuführen. Ein weiteres Ziel der CLDC ist es, die Java-Plattform nicht nur aufgrund der eingeschränkten Ressourcen zu minimalisieren, sondern gleichzeitig einen mächtigen Befehlssatz für die möglichst einfache Entwicklung mobiler Anwendungen bereitzustellen. Wie die CLDC diesen Spagat zwischen der Einschränkung der Komplexität und der Beibehaltung einer möglichst großen Kompatibilität zur Java 2 Standard Edition schafft, wird in diesem Kapitel näher beschrieben.

2.1 Die Spezifikation der CLDC

In der Spezifikation der CLDC existiert keine Forderung nach einer eingeschränkten Hardwareausstattung, da sehr unterschiedliche Geräte für die Java-Plattform in Frage kommen. Die einzige Limitierung bezieht sich auf den Bedarf an Speicher, den das Gerät bereitstellen muss und auf die Vernetzung des Gerätes:

- Es müssen mindestens 160 Kilobyte nichtflüchtiger Speicher (ROM) für die Java-Umgebung zur Verfügung stehen. Diese werden von der Virtual Machine und den CLDC Bibliotheken in Anspruch genommen.
- Es müssen mindestens 32 Kilobyte flüchtiger Speicher (RAM) für die Laufzeitumgebung von Java bereitgestellt werden, das heißt für die zu CLDC gehörende Virtual Machine.
- Die Verbindung zu einem Netzwerk muss vorhanden sein. Hierbei kann die Verbindung wahlweise drahtlos erfolgen. Die Bandbreite darf zudem unter 9600 Baud liegen.
- Das Endgerät darf darüber hinaus über eine mobile Stromversorgung verfügen.

Der Anteil von flüchtigem und nichtflüchtigem Speicher in Bezug auf das Gesamtbudget des Speichers kann variieren, je nachdem welche Aufgaben die Java-Plattform auf dem Zielsystem zu erfüllen hat. Falls die Java-Plattform nur für fest installierte Systemanwendungen genutzt wird, ist nur ein sehr geringer Anteil flüchtigen Speichers notwendig. Wenn die Java-Plattform dagegen für die Ausführung dynamischen Inhalts per Netzwerkverbindung genutzt wird, dann benötigt das verwendete Endgerät wesentlich mehr flüchtigen Speicher als angegeben.

Entsprechend der Hardwarevoraussetzungen, sind auch die Anforderungen an das Betriebssystem des verwendeten Endgerätes relativ gering und definieren nur die Mindestvoraussetzung. Die CLDC Spezifikation sieht vor, dass nur ein sehr minimales Betriebssystem vorhanden sein muss, zur Steuerung der darunter liegenden Hardware. Es wird keine Unterstützung von getrennten Adressräumen von dem Betriebssystem gefordert, ebenso wenig wie die Ablaufsteuerung mehrerer gleichzeitiger Prozesse. Es muss nur die Ablaufsteuerung eines einzelnen Prozesses überwacht werden, um die Java Virtual Machine ausführen zu können.

2.2 Aufgaben der CLDC

Der Aufgabenbereich der CLDC umfasst folgende Punkte:

- Festlegung der Java Sprache und die Merkmale der Virtual Machine
- Definition des Kerns der Java Bibliotheken [`java.lang.*`, `java.util.*`]
- Eingabe und Ausgabe auf dem Endgerät [`java.io.*`]
- Vernetzung
- Sicherheit
- Internationalisierung

Folgende Punkte zählen nicht zum Aufgabenbereich der CLDC Konfiguration, sondern werden von den jeweiligen Profilen umgesetzt:

- Installation, Ausführung und Entfernen von Anwendungen
- Benutzerschnittstelle
- Ereignisbehandlung

2.3 Klassen der CLDC

Die Java 2 Standard Edition besitzt sehr umfangreiche Klassen-Bibliotheken für die Entwicklung von Desktop-Anwendungen. Da diese Bibliotheken etliche Megabyte an Speicher beanspruchen, sind sie vollkommen ungeeignet für die kleinen, eingeschränkten Geräte innerhalb der CLDC-Spezifikation. Wie bereits erwähnt, ist das Hauptproblem bei der Festlegung der Bibliotheken für die CLDC-Spezifikation, einerseits die Größe möglichst zu minimieren und andererseits die Entwicklung von Anwendung und die Definition von Profilen weiterhin bestmöglich zu unterstützen. Der Lösungsansatz, den die CLDC Spezifikation für dieses Problem anbietet, wird häufig als „kleinster gemeinsamer Teiler“ bezeichnet, da dieser Standard nur minimale Merkmale der Java-Plattform besitzt und dennoch APIs für ein großes Spektrum unterschiedlichster Endgeräte bereitstellt. Bei diesen starken Einschränkungen einerseits und der Produktvielfalt andererseits, ist dies jedoch die einzige Möglichkeit auf einen gemeinsamen Nenner zu kommen, da man ansonsten immer den einen oder anderen Gerätetyp benachteiligen würde.

Um Aufwärtskompatibilität zu größeren Java-2-Plattformen zu gewährleisten, sind die meisten in CLDC integrierten Klassen-Bibliotheken eine Teilmenge der J2SE-Klassen. Diese Art von Vererbung funktioniert bei den meisten Klassen, bei einigen ist es jedoch aufgrund innerer Abhängigkeiten nicht möglich, die Klassen noch weiter zu vereinfachen und in die J2ME zu übertragen. Diese inneren Abhängigkeiten, die teilweise auch eine bestimmte Hardware voraussetzen, sind insbesondere bei wichtigen Themen wie Sicherheit, Ein-/Ausgabe, Benutzerschnittstelle, Speicherzugriff und Vernetzung vorhanden. In diesen Fällen müssen die Klassen-Bibliotheken speziell an die CLDC angepasst werden. Dabei gilt es den Grundsatz der Aufwärtskompatibilität zu beachten, dass J2ME-Klassen mit dem gleichen Namen wie in J2SE keine zusätzlichen Methoden enthalten dürfen.

Die CLDC ist in so genannte Packages untergegliedert, welche die Klasseneinteilung vornehmen. Es gibt vier Packages, die man aufgrund der enthaltenen Klassen in zwei Kategorien einteilen kann:

- i. **Klassen, die nur eine Teilmenge der J2SE-Bibliotheken darstellen:**
 - `java.io`: Implementiert Klassen zur Ein- und Ausgabe durch DataStreams.
 - `java.lang`: Enthält wie in J2SE die fundamentalen Klassen von Java.
 - `java.util`: Enthält einige nützliche Klassen wie die Datums- und Zeitbehandlung.
- ii. **Klassen, die CLDC-spezifisch sind:**
 - `javax.microedition.io`

2.3.1 CLDC-spezifische Klassen

Die Java 2 Standard Edition unterstützt viele unterschiedliche Arten der Ein- und Ausgabe im Bereich des Netzwerkzugriffes und des Speicherzugriffes. Um dies zu ermöglichen existiert eine große Anzahl an Funktionen für den Verbindungsaufbau. Diese Funktionen befinden sich in den J2SE-Klassen-Bibliotheken und sind in diesem Umfang nicht in die CLDC übertragbar, da dies zu viel Speicherplatz beanspruchen würde. Aus diesem Grund ist in dem Package „`javax.microedition.io`“ eine Reihe von Verbindungsklassen unter dem Namen **Generic Connection Framework (GCF)** erfasst. Diese Klassen bilden, wie der Name bereits vermittelt, einen allgemeinen Standard für den Aufbau von Verbindungen. Durch diesen Standard war es möglich, den Zugriff auf alle vorhandenen Schnittstellen einheitlich zu realisieren. Im Vergleich zu dem bisherigen Modell, bei dem man für jede Verbindungsart unterschiedliche Realisierungen und Datentypen in Betracht ziehen musste, stellt

dieser Ansatz eine wesentliche Erleichterung dar. Um bei einem Verbindungsaufbau das richtige Protokoll für die Datenübertragung auszuwählen, übergibt man der Methode „`Connector.open`“²⁾ einen Parameter in Form eines Strings. Ist der Verbindungsaufbau erfolgreich, gibt die Methode ein Objekt zurück, das eines der untergeordneten Interfaces ausführt. Die CFG bietet insgesamt sieben so genannte Interfaces für die unterschiedlichen Aufgaben des Datenverkehrs. Wie in Abbildung 2 zu sehen, sind die Interfaces hierarchisch angeordnet und beginnen mit dem „`Connection`“-Interface. Der Syntax eines „`Connector.open`“-Aufrufes stellt sich wie folgt dar:

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

Es folgt ein Beispiel für einen typischen „`Connector.open`“-Aufruf, um eine Verbindung vom Typ `Connection` als Rückgabewert zu erhalten:

```
(URLConnection)Connector.open("http://www.sun.com:port");
```

Ein weiteres Beispiel ist das Anfordern einer seriellen Verbindung mit festgelegter Baudrate:

```
Connector.open("comm:0;baudrate=9600");
```

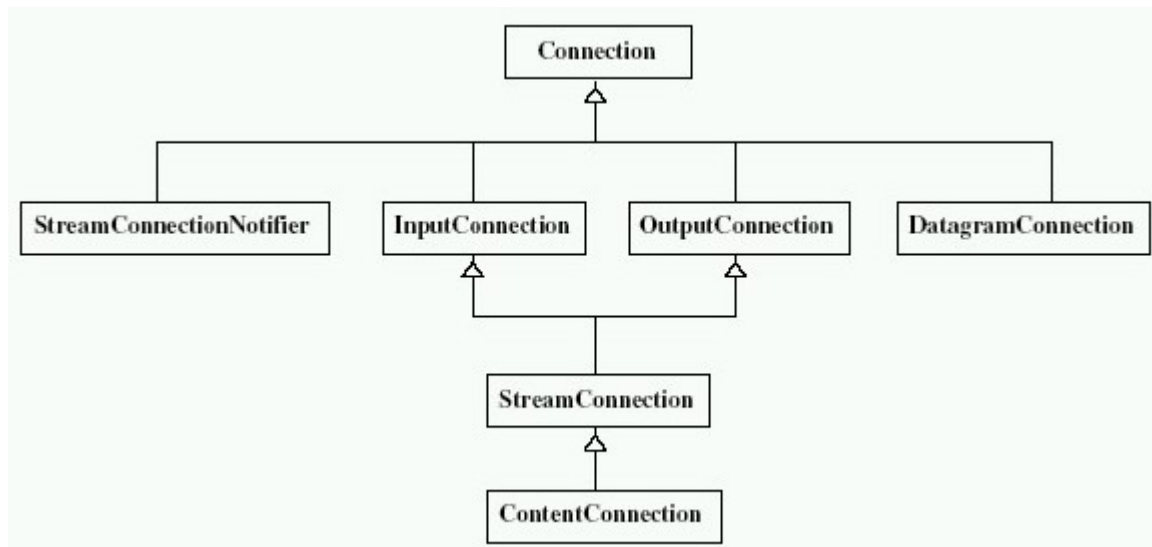


Abbildung 2: Hierarchie der „`Connection`“-Interfaces

2.4 Das Sicherheitssystem der CLDC

In unserer heutigen Zeit wird die Frage der Sicherheit von kritischen Informationen mehr und mehr den Computersystemen überlassen. Nicht nur im Büro, sondern auch im Privatleben steht das zuverlässige Absichern der Daten vor unerwünschten Eindringlingen hoch im Kurs. Das Thema Sicherheit ist im Bereich mobiler Anwendungen und kabelloser Netze ein besonders kritischer Punkt, auf den es zu achten gilt. Die Java 2 Standard Edition ist in diesem Gebiet bestens gerüstet und unterstützt die Entwickler mit einer Reihe mächtiger und flexibler Schutzmechanismen. Die Java 2 Micro Edition unterliegt jedoch der angesprochenen Ressourcenbeschränkung, weswegen das Sicherheitsmodell der CLDC wesentlich größere Einschränkungen zu Gunsten gesteigerter Sicherheit bietet. Das Sicherheitsmodell der CLDC ist in zwei wichtige Ebenen eingeteilt:

²⁾ Die Methode „`Connector.open`“ gehört zu der Klasse „`javax.microedition.Connector`“.

2.4.1 Das Sicherheitssystem auf unterster Ebene

Damit die Java Virtual Machine auf einem Endgerät korrekt funktioniert, ist ein Sicherheitssystem auf unterster Ebene notwendig. Dieses System sorgt dafür, dass auf dem Gerät ausgeführte Anwendungen dieses nicht zum Absturz bringen oder gar Schaden zufügen können. In der J2SE wird dies durch eine so genannte Klassenverifikation garantiert, welche den Befehlscode der Klassendateien nach unerlaubten Operationen oder falschen Referenzen überprüft. Diese Überprüfung beinhaltet einige komplexe Algorithmen und beansprucht einiges an Prozessorleistung und Speicherausstattung, und ist deswegen für die Geräte der CLDC völlig ungeeignet. Als Alternative wurde die so genannte **Pre-verifikation** entwickelt, die mit deutlich weniger Systemressourcen auskommt und den gesamten Verifikationsprozess in zwei Teile unterteilt (der komplette Vorgang ist in Abbildung 3 dargestellt):

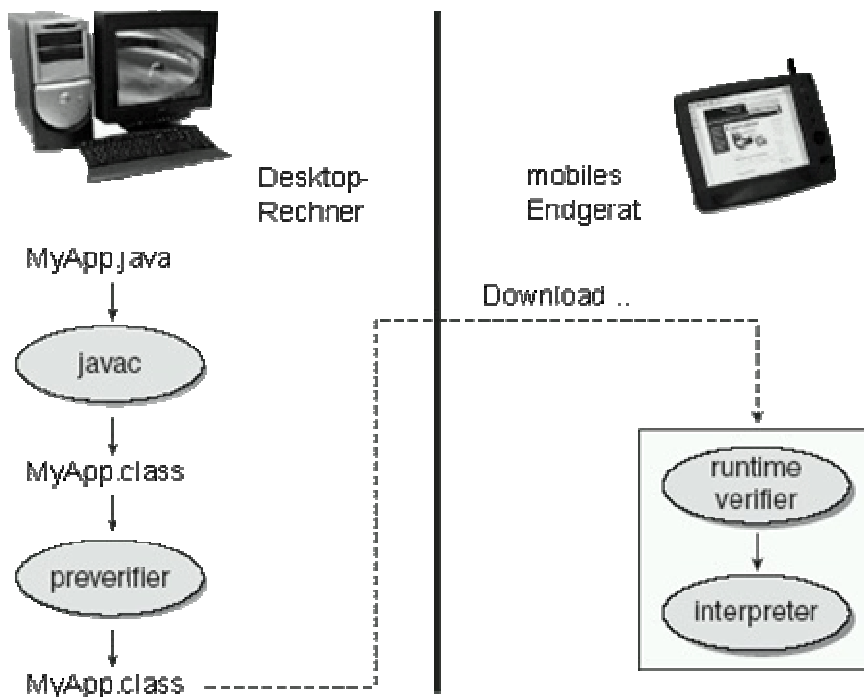


Abbildung 3: Der Ablauf der Pre-verifikation in der CLDC

- i. Die eigentliche Klassenverifikation wird auf den Desktop-Rechner des Entwicklers durchgeführt. Der so genannte **preverifier** überprüft die Klassendateien und markiert die korrekten Dateien mit einem Attribut, das sich „StackMap“ nennt. Dieses Attribut vergrößert die damit gekennzeichneten Klassendateien um ungefähr 5%, außerdem wird es in der J2SE-Umgebung ignoriert – die Aufwärtskompatibilität ist somit gewährleistet.
- ii. Auf dem mobilen Endgerät findet zur Laufzeit erneut eine Klassenverifikation durch den so genannten **runtime verifier** statt. Diese Überprüfung beansprucht sehr wenige Systemressourcen, da nur das „StackMap“-Attribut überprüft wird und nicht die kompletten Klassendateien. Der Befehlscode von Klassendateien wird erst ausgeführt, wenn die Klassen die Laufzeitüberprüfung bestanden haben.

Diese Lösung bietet denselben Sicherheitsstandard wie J2SE, benötigt jedoch deutlich weniger Systemleistung auf dem mobilen Endgerät.

2.4.2 Das Sicherheitssystem auf der Ebene der Anwendungen

Obwohl die Klassenverifikation eine große Rolle innerhalb des Java-Sicherheitssystems spielt, kann es nicht alle Sicherheitslücken beseitigen. Zwar sorgt es dafür, dass nur gültige Java-Anwendungen ausgeführt werden, doch es hat keinen Einfluss auf die von den Anwendungen durchgeführten Aktionen. Ein weiteres Sicherheitssystem muss die Anwendungen überwachen und dafür sorgen, dass nur auf die freigegebenen Klassenbibliotheken und Systemressourcen zugegriffen wird. Um dies zu gewährleisten, besteht die Ebene der Anwendungen aus unterschiedlichen Sicherheitssystemen:

- i. Bereits bei der Entwicklung der Java-Sprache wurde besonderer Wert auf Sicherheit gelegt und das Prinzip des **Sandbox**-Modells entworfen. In diesem Modell wird eine feste Laufzeitumgebung (Sandbox) zur Verfügung gestellt, innerhalb der die Applikationen eine gesicherte Ablaufumgebung finden. Java-Anwendungen können nicht aus diesem „Sandkasten“³⁾ ausbrechen, sondern dürfen nur auf Klassenbibliotheken oder Ressourcen zugreifen, die zuvor definiert wurden. Dieses Sicherheitsmodell ist bei der CLDC zu Gunsten gesteigerter Sicherheit eingeschränkt, das heißt man hat keine Möglichkeiten, die Laufzeitumgebung an die eigenen Bedürfnisse anzupassen.
- ii. Eine zentrale Anwendung von CLDC ist die Durchführung eines dynamischen Downloads einer Java-Applikation auf das mobile Endgerät. Bei diesem Vorgang wird ein weiterer Sicherheitsmechanismus notwendig: Der Schutz der Systemklassen. Dieser Mechanismus verhindert, dass neu installierte Anwendungen Systemklassen überschreiben.

2.5 Weitere Einschränkungen der CLDC

Im Vergleich zur J2SE fehlen den Klassenbibliotheken der CLDC einige Funktionen, aufgrund der stark eingeschränkten Ressourcen der mobilen Kommunikationsgeräte. Abseits der bisher beschriebenen Themen, gibt es eine Reihe weiterer Punkte, die zu erwähnen sind:

2.5.1 Fließkommazahlen

Die Berechnung der Fließkommazahlen ist sehr rechenintensiv und meist an eine spezielle Hardware gebunden. Der Hauptprozessor kann somit wesentlich entlastet werden und seine Ressourcen für andere Dienste zur Verfügung stellen. Bei den meisten Mobilgeräten ist allerdings weder diese spezielle Hardware vorhanden, noch die Taktfrequenz des Prozessors ausreichend für die Berechnung der Fließkommazahlen. Aus diesem Grund sieht die Spezifikation der CLDC keine Fließkommaunterstützung vor, was zur Folge hat, dass die Datentypen `float` und `double` ebenso wenig vorhanden sind wie die dazugehörigen Methoden zur Berechnung.

2.5.2 finalize-Methode

Ein weiterer Schritt zur Reduzierung des Ressourcenverbrauchs bei CLDC-Geräten ist die fehlende Möglichkeit, einer Klasse die `finalize()`-Methode hinzuzufügen. Die `finalize()`-Methode der J2SE-Plattform dient innerhalb einer Methode üblicherweise als Mittel zum sicheren Schließen offener Verbindungen (z.B. zum Speicher, Grafikkontext oder zu Streams).

2.5.3 Fehler und Exceptions

Auch die Fehlerbehandlung bei mobilen Endgeräten fällt nicht so umfangreich aus wie bei normalen Desktop-Systemen. Dies wird bereits bei den Exceptions deutlich, die im wesentlichen nur sehr eingeschränkt implementierten wurden, asynchrone Exceptions fehlen beispielsweise komplett. Zudem ist die Fehlerbehandlung bei den betreffenden Gerätegruppen oftmals dadurch gelöst, dass die Hardware durch einen Reset in den Normalzustand zurückversetzt wird.

³⁾ In Analogie zu einem Kind, das sich in seinem Sandkasten befindet und dort keinen Schaden anrichten kann.

2.5.4 Class-Loader

Der Class-Loader steuert gewöhnlich den Zugriff auf die Klassen und lädt die dazugehörigen Bibliotheken nach den Vorgaben der `CLASSPATH`-Variablen. Außerdem ermöglicht das Class-Loader-Konzept der J2SE weit reichende Anpassungsmöglichkeiten bei der Funktionsweise. Diese Eingriffsmöglichkeit des Benutzers wurde aufgrund höherer Sicherheitsrichtlinien der CLDC nicht realisiert.

3. Die CDC im Detail

3.1 Zielgruppe der CDC

Die Connected Device Configuration wurde für Endgeräte entwickelt, die eine bessere technische Ausstattung als die Geräte der CLDC besitzen. Dabei handelt es sich um Geräte, die eine größere Speicherausstattung, einen schnelleren Prozessor und eine stabilere Netzwerkanbindung besitzen, aber nicht die volle Leistungsfähigkeit eines Desktop-Systems. Auch ein CDC-Gerät besitzt also limitierte Hardwareressourcen, die jedoch etwas leistungsstärker sind:

- Das Endgerät muss mindestens einen 32-bit Prozessor besitzen.
- Es müssen mindestens 2 Megabyte nichtflüchtiger Speicher (ROM) für die Virtual Machine und die CDC Bibliotheken zur Verfügung stehen.
- Es müssen mindestens 2 Megabyte flüchtiger Speicher (RAM) für die Laufzeitumgebung von Java bereitgestellt werden.
- Die Verbindung zu einem Netzwerk muss vorhanden sein. Hierbei kann die Verbindung wahlweise drahtlos erfolgen. Die Bandbreite darf zudem unter 9600 Baud liegen.

3.2 Einschränkungen der CDC

Die Hardwareressourcen der CDC-Geräte sind ausreichend dimensioniert, so dass die Virtual Machine der J2SE nicht eingeschränkt werden muss, sondern die komplette Spezifikation unterstützen werden kann. Dazu gehören im Gegensatz zu der CLDC auch Eigenschaften wie beispielsweise benutzerspezifische Class-Loader, eine vollständige Fehlerbehandlung oder die Unterstützung von Fließkommazahlen. Auch die Klassen-Bibliotheken der CDC weichen bei weitem nicht so stark von dem J2SE-Standard ab, wie bei der CLDC. Die einzige Einschränkung der J2SE-Spezifikation betrifft die in den Klassen-Bibliotheken vorhandenen Schnittstellen, hier wurden nur die benötigten Schnittstellen beibehalten und den Anforderungen angepasst. Die übrigen Klassen-Bibliotheken der CDC entsprechen denen der J2SE, sie wurden allerdings für den Betrieb in Geräten mit einer geringen Speicherausstattung optimiert. Die auf diese Weise entstandene Java-Plattform funktioniert perfekt mit einem Speicherbudget von 2 MB RAM und 2MB ROM.

3.3 Aufbau der CDC

Die CDC ist sehr modular aufgebaut, wobei die Profile der CDC einen hohen Stellenwert einnehmen. Jede Implementation eines CDC-Profiles beinhaltet die Realisierung einer Klassen-Bibliothek und einer Java Virtual Machine. Die CDC-Profile sind in mehreren Schichten angeordnet und können vielfältig kombiniert werden. Auf diese Weise lassen sich Funktionalitäten hinzufügen oder entfernen und optimal an die jeweilige Hardwarebasis anpassen. Die unterste Ebene der Profile bildet das **Foundation Profile**. Das Foundation Profile besteht aus einigen Java-APIs und einer kompletten J2ME Laufzeitumgebung, die speziell auf die so genannten **embedded devices**, die Endgeräte der CDC zugeschnitten ist. Es unterstützt insbesondere die Netzwerkfunktionalität dieser Geräte, eine Benutzerschnittstelle ist jedoch nicht vorgesehen. Um diesen Nachteil auszugleichen, kann man bei Geräten mit grafischer Benutzerschnittstelle (GUI) das **Personal Basis Profil** und das **Personal Profil** als nachfolgende Profilschichten installieren.

3.4 Das Sicherheitssystem der CDC

Die Sicherheit ist ein zentrales Grundelement der Java-Programmiersprache. In der Spezifikation der CDC sind fünf Sicherheitsebenen vorgesehen, die sowohl dem Benutzer als auch dem Anwendungsentwickler eine große Auswahl mächtiger Sicherheitsmechanismen zur Unterstützung anbieten:

3.4.1 Virtual Machine Security

Die Sicherheitsebene der Virtual Machine ist die unterste Schutzebene des Sicherheitskonzeptes. Darin enthalten sind eine **Klassenverifikation**, die im Gegensatz zur CLDC von der J2SE-Spezifikation übernommen wurde, und verschiedene Spracheigenschaften. Zu diesen Eigenschaften gehört zum Beispiel das Entfernen von **Pointer** aufgrund zu hoher Sicherheitsrisiken.

3.4.2 Signed Classes

Diese so genannten „gekennzeichneten Klassen“ stellen eine Verbesserung des aus der CLDC bekannten Sandbox-Modells dar. Sie garantieren die Vollständigkeit und Korrektheit der Java Klassendatei, die anschließend von der Virtual Machine geladen wird.

3.4.3 Policy-based security

Dieses Sicherheitssystem basiert auf Richtlinien und unterstützt die Softwareentwickler mit einer fein abgestuften Kontrollfunktion. Mit dieser Kontrollfunktion kann man festlegen, welchem Benutzer man den Zugriff auf Daten, Schnittstellen oder Anwendung erlauben will. Diese Richtlinien werden von dem Softwareentwickler festgelegt und können nur von dem Systemadministrator angepasst werden.

3.4.4 Cryptography

Verschlüsselung bietet die Möglichkeit, Software und Daten für eine sichere Übertragung zu chiffrieren. Bei dem Java Sicherheitssystem ist die Verschlüsselung durch die so genannte **Java Cryptography Architecture (JCA)** realisiert.

3.4.5 Certificate management

Bei diesem standardisierten Mechanismus findet ein Authentifizierungsprozess statt, der durch ein Zertifikat einer Person den Zugriff gewährt.

4. Die Configurations im Überblick

4.1 Einschränkungen bei der CLDC im Vergleich zum J2SE-Standard

- Eingeschränkte Klassen
- Klassenverifikation = Pre-Verifikation
- Andere Sicherheitsmechanismen
- Keine Fließkommazahlen
- Eingeschränktes Fehler- / Exceptionhandling
- Keine finalize-Methode
- Keine angepassten Class-Loader

4.2 Einschränkungen bei der CDC im Vergleich zum J2SE-Standard

- Keine Schnittstellen bei den Klassen-Bibliotheken
- Hardwareoptimierte Klassen-Bibliotheken
- Andere Sicherheitsmechanismen

4.3 Tabellarische Gegenüberstellung

CLDC	CDC
Endgerät wird von 16/32-Bit CPU betrieben	Endgerät wird von 32-Bit CPU betrieben
Mindestens 160 KB dauerhaften Speicher	Mindestens 2 MB dauerhaften Speicher
Mindestens 32 KB Speicher für die Laufzeitumgebung	Mindestens 2 MB Speicher für die Laufzeitumgebung
Instabile Netzwerkverbindung	Stabile Netzwerkverbindung
Netzwerkverbindung basiert nicht auf TCP/IP	Netzwerkverbindung basiert auf TCP/IP
Endgerät darf mobile Stromversorgung haben	Endgerät hat keine mobile Stromversorgung

Eine Verbindung zum Netzwerk muss vorhanden sein, eventuell auch drahtlos.
Die Bandbreite darf unter 9600 Baud liegen.

5. Abbildungsverzeichnis

Abbildung 1:	Struktur der Java-ME-Plattform	2
	Eigenkreation nach Beispielen im CLDC-Whitepaper von Sun Microsystems	
Abbildung 2:	Hierarchie der "Connection"-Interfaces	7
	Quelle: CLDC Specification Version 1.1 von Sun Microsystems, März 2003	
Abbildung 3:	Der Ablauf der Pre-verifikation in der CLDC	8
	Quelle: CLDC Specification Version 1.1 von Sun Microsystems, März 2003	

6. Literaturverzeichnis

- | | |
|---|--|
| [1] J2ME Building Blocks for Mobile Devices
http://java.sun.com/products/cldc/wp/KVMwp.pdf | Technical White Paper
Sun Microsystems, Inc.
19. Mai, 2000 |
| [2] CLDC Version 1.1
http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html | Specification
Sun Microsystems, Inc.
März, 2003 |
| [3] CLDC HotSpot™ Implementation, Virtual Machine
http://java.sun.com/products/cldc/wp/CLDC-HI_whitepaper-March_2004.pdf | Technical White Paper
Sun Microsystems, Inc.
März, 2004 |
| [4] CDC: An Application Framework for Personal Mobile Devices
http://java.sun.com/products/cdc/wp/cdc-whitepaper.pdf | Technical White Paper
Sun Microsystems, Inc.
Juni, 2003 |
| [5] CDC and the Foundation Profile
http://java.sun.com/products/cdc/wp/CDCwp.pdf | Technical White Paper
Sun Microsystems, Inc.
2001 |