

Hauptseminar (WS 2001/2002)

Einführung in die Java Card Technologie

Nachweise von Sicherheitseigenschaften für JavaCard
durch approximative Programmauswertung

Veranstalter Prof. T. Nipkow
Dr. M. Strecker

Autor Tao Zhuang (zhuangt@in.tum.de)

07 , Januar , 2002

Inhaltsverzeichnis

- 1 Einleitung
- 2 Smart Card Basis
 - 2.1 Überblick der Smart Card
 - 2.2 Verschiedene Art von Smart Card
 - 2.2.1 Memory Cards im Vergleich zur Mikroprozessor Cards
 - 2.2.2 Kontakte Karte im Vergleich zur nonkontakte Karte
 - 2.3 Smart Card Hardware
 - 2.4 Smart Card Kommunikation
 - 2.5 Smart Card Operationssystem
 - 2.6 Smart Card System
- 3 Java Card Technologie
 - 3.1 Überblick der Architektur
 - 3.2 Java Card Sprachesubmenge
 - 3.3 JCVM
 - 3.4 Java Card Installer und Off_Card Installationsprogramm
 - 3.5 JCRE
 - 3.6 Java Card APIs
 - 3.7 Java Applet Entwicklungsprozess
 - 3.8 Applet Installation
- 4 Objekte in Java Card Technologie
 - 4.1 Java Card Memory Model
 - 4.2 Persistent Objekte
 - 4.3 Transiente Objekte
 - 4.3.1 Eigenschaft der transient Objekte
 - 4.3.2 Transienter Objekte Type
 - 4.3.3 Erzeugung der transienten Objekte

5 Atomisierung und Transaktionen

5.1 Atomicity

5.2 Block Data Update in einem Array

5.3 Transaktion

5.3.1 Commit Transaktion

5.3.2 Stoppen Transaktion

5.3.3 Vernetzte Transaktion

5.3.4 Commit Kapazität

5.3.5 TransaktionException

5.3.6 Lokale Variable und transiente Objekte während einer Transaktion

1 Einleitung

Seit langer Zeit hat die Industrie eine Technologie gesucht , die sich an die Zuverlässigkeit und Sicherheit der Forderung von Industrie anpasst . Endlich ist es Zustand gekommen , dass in 1988 die erste kommerzielle Smart Card auf die Markt erscheite , wenn News Group's BskyB (Nämlich SkyTV) in England a neue Generations- Card für ihre innovative Satellit Pay-Television System brauchte .

Damals stellte nur assemble Programmierungssprache für die Softwareentwickler zur Verfügung .Nachdem die Java Programmierungssprache erscheite , ist die Java Card Technologie außergewöhnlich marschiert .Mit der Eigenschaft von Java Programmierungssprache ist die Zuverlässigkeit und Sicherheit von Java Card Technologie garantiert . Im Laufe der Zeit ist die Java Card Standard mit der Unterstützung von der Firma Sun entwickelt . Die Java Card ist eine mit Chip versehene Plastikkarte . Die ist ein tragbares Computer, der Prozessenergie und Information besitzt , nämlich der keinen Zugang zur Fernkontrolle des Datenbank während der Transaktion braucht . Deshalb kann Java Card in vielen Bereichen verwendet (z.B. Bezahlung und Banking Industrie , Gesundheitsbesorgung , Internet ,in einer geschlossenen Umgebung) werden . Äußerordentliche Eigenschaft von Java Card ist, die Applikationen bzw. Applets kann nach der Anfertigung der Karte einladen werden . Smart Card ist oft zur Speicherung der Sicherheitscode nämlich PIN (personal identification number) , das der Kunde um sich zu identifizieren braucht .Damit erhalten die Kunden automatisch eine E-Purse (oder ein e-Geldbörse) . Mit dieser Karte braucht man nur die Karte in Automat einzustecken und auszuziehen nachdem die Prozedur erledigt ist .

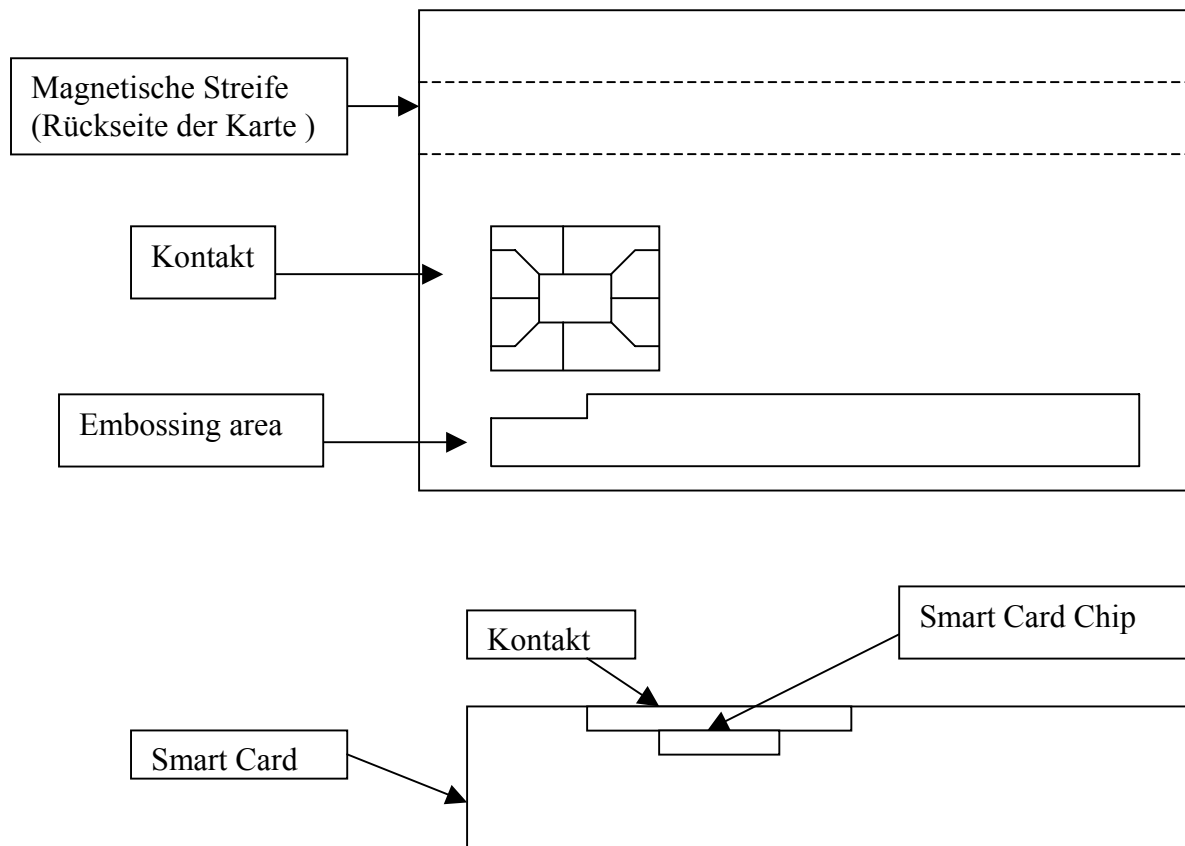
Java Card Technologie bietet eine Sichere , tragbare und Multi-applikation Smart Card Betriebssystem . Vorteile von Java Card Technologie sind folgende :

1. Leicht für Applikationsentwicklung wegen der geöffnete Betriebsumgebung und Standardinterface.
2. Sicherheit wegen der Eigenschaft von Java Programmiersprache .
3. Hardware Unabhängigkeit wegen der Java Card Technologie
4. Fähigkeit zu Speicherung und Kontrolle der Applikationen

2. Smart Card Basis

2.1 Überblick der Smart Card

Smart Card , sogenannte Chip Karte oder IC (integrated circuit) Karte , wird für Datentransmission ,Ablegen verwendet .Eine Smart Card sieht so aus.



2.2 Verschiedene Art von Smart Card

Die Smart Card unterscheidet sich von Memory Card und Mikroprozessor Card .Die Smart Card kann sich von Kontakt Card und kontaktfreie Card unterscheidet .

2.2.1 Memory Cards im Vergleich zur Mikroprozessor Cards

Die frühest produzierte Smart Card ist Memory Card , die nicht echte Smart Card ist , weil die keine Mikroprozessor besitzt und mit a Memory Chip und einem Chip mit Memory und unprogrammierbare Logik versehen ist . Ein Nachteil von Memory Card ist , die nur 1K bis 4K Daten speichern kann .Ein anderer Nachteil von Memory Car ist ,die nicht wiederverwendbar ist, da die kein CPU hat , um die Daten zu bearbeiten , genau gesagt , die kann nur einige vorprogrammierte Befehle durchführen .Der Vorteil ist jedoch wesentlich , da die von Memory Card verwendete Technologie ziemlich einfach im Vergleich zur Mikroprozessor Card , deshalb ist die Memory Card gefragt in Low-cost Card Markt.

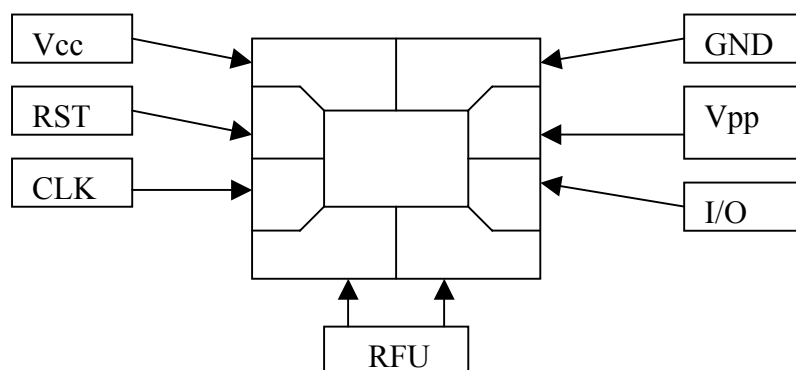
Gegenüber Memory Card besitzt die Mikroprozessor Card einen Prozessor ,der mehrere Sicherheit und multifunktionale Kapazität
Anbietet . Der Mikroprozessor kontrolliert Datenbearbeitung und Speicherzugreifen und zwar durch die Angabe (z.B Password). Die kann sowohl in einer Applikation als auch in mehreren Applikationen angewendet werden, weil die sehr flexibel ist .

2.2.2 Kontakte Karte im Vergleich zur nonkontakt Karte

Kontakt Karte muss in eine Geräte eingesteckt werden ,damit die sich mit dem Außenwelt kommuniziert .Dagegen braucht die nonkontakt Karte keine Geräte , weil die mit Außenwelt mit einem in der Card liegender Antenne . Die Kommunikation zwischen der Karte und einer Card Akzeptanzgeräte ist durch die elektromagnetische Signal ausgeführt . Deswegen ist die in der Beriechen von schneller Transaktion favorisiert . Ein Vorteil von nonkonakte Karte ist , es gibt keine Benutzungsgrenze für die Karte nämlich die kann sehr oft benutzt werden . Ein Nachteil davon ist , man muss eine gewisse Abstand zwischen der Karte und der Akzeptanzgeräte halten um die Datentransport zu ermöglichen .Ein Prozedur wird abgebrochen ,falls der Abstand zu wenig und die Karte zu schnell vor der Geräte überspringt .

2.3 Smart Card Hardware

Eine Smart Card Chip sieht wie folgende aus .



Eine Smart Card hat achte Kontaktpunkte .

1. Vcc bietet der Betriebsstrom in der Höhe von 3-5 Volt für die Karte an.
2. RST sogenannt Warm Reset . Der sendet die Signal zu den Mikroprozessor . Ein Cold Reset ist , wenn der Betriebsstrom ausgeschaltet und wieder eingeschaltet ist (z.B die Karte wird ausgezogen und wieder eingesteckt .).
3. CLK bietet Zeitsignal für die Karte.
4. GND ist Basisvolt meistens ist es 0 Volt .
5. Vpp ist für die alte Modelle verfügbar.
6. I/O ist für Datentransport entworfen .Das Transport kann nur bei einer Richtung durchgeführt und zwar in einem gewissen Zeitraum .
7. RUF ist für die künftige Funktionen entworfen .

Der zentrale CPU war einen 8-Bit Mikrokontroller . Die neue Smart Card besitzt 16 oder 32-Bit Mikrokontroller . Die neue Smart Card besitzt normalerweise auch einen Koprozessor ,der für die spezielle Berechnung zuständig ist und den Produktkost wesentlich beeinflusst .

Es gibt drei Arte von Speichern :

- 1 Rom (read –only momory) : Die konstante Programme werden darin speichert .Nachdem die anfertigt ist , ist die nicht mehr zu schreiben und nur zu zugreifen .
- 2 EEPROM (Eletrical erasable programmable read memory):
Man kann die Daten auch darauf speichern .Nachdem der Betriebsstrom ausgeschaltet ist , ist die gespeicherten Informationen noch gebelieben .Die Unterschiede zwischen Rom und EEPROM ist , die Daten kann noch auf EEPROM gespeichert werden , nachdem die Karte anfertigt ist .
- 3 RAM (random access memory):ist ein vorläufiger Speicher . Nachdem der Betriebsstrom ausgeschaltet ist , ist die gespeicherten Informationen ewig verloren .

Nach dem Produktkost ist ROM der teuerste dahinfolegende ist EEPROM und RAM .

2.4 Smart Card Kommunikation

CAD , sogenannte Card Acceptance Device , vermittelt die Information von der Karte zu dem Computer . CAD unterscheidet sich in Zwei Arten .

- 1 Leser : Der ist angeschlossen mit der Computer .Mit dem Leser kann die Information von der Karte zum Hostcomputer weitergegeben werden .Ein Leser hat keine Intelligenz , die Daten zu überarbeiten , manchmal passiert die Fehler , falls die transportierten Daten und falschem Protokoll mitgeliefert werden .
- 2 Terminal :Der ist ein echter Computer .Der kann die Daten überarbeiten , deshalb ist Terminal in vielen Bereichen verwendet (z.B Tankstelle , Transaktion der Kreditkarte , Bank ATM)

Die Kommunikation zwischen der Karte und dem Host ist immer in einer Richtung durchgeführt ,d.h. die Datei ist entweder von der Karte zu dem Host oder umgekehrt und sie das in Internet verwendete Protokoll TCP/IP wird APDUs (Application protocol data unit) für Smart Card Kommunikation angewendet .APDUs ist ein Applikationsstufe Protokoll zwischen einer Smart Card und einem Hostcomputer . Ein APDUs besteht aus zwei Strukturen .

- 1 Command APDU : Host sendet CAD die Befehle.
- 2 Response APDU : Die Card sendet dem Host die Information zurück .

Die Struktur des APDUs ist wie folgende definiert .

Mandatory header				Optional Body		
CLA	INS	P1	P2	Lc	Data Field	Le

Befehl APDU Struktur

Optional Body	Mandatory Trailer	
Data Field	SW1	SW2

Response APDU Struktur

CLA : Instruktionsklass

INS: das Kode der Instruktion

P1, P2 :Parameter 1 und 2

Lc :Länge der Befehle

Data field :zu Card gesendete Daten um die Transaktion auszuführen.

Le: die Anzahl der von der Card zum Host zurückgesendete Befehle in Byte

Die Response APDU besteht aus zwei Teilen :

- 1 Optional Body :Die Länge des Optionale Body ist durch Lc vordefiniert.
- 2 Mandatory Trailer : Zwei Teile SW1 und SW2 .Die gibt die Resultat nach Ausführung des Befehls .(z.B 0x9000 bedeutet , ein Befehl ist erfolgreich und total ausgeführt .

Es gibt vier Fälle für APDU:

- 1 Keine Daten werden von oder zu Card gesendet , nämlich Befehl APDU hat nur Header und Response APDU hat nur das Traliler Status Word
- 2 Keine Daten werden zu Card transportiert .Jedoch sendet Card Daten zum Host .
- 3 Die Daten werden zu Card transportiert .Jedoch sendet Card keine Daten zum Host .
- 4 Die Daten werden zu Card transportiert , und sendet Card die Daten zum Host .

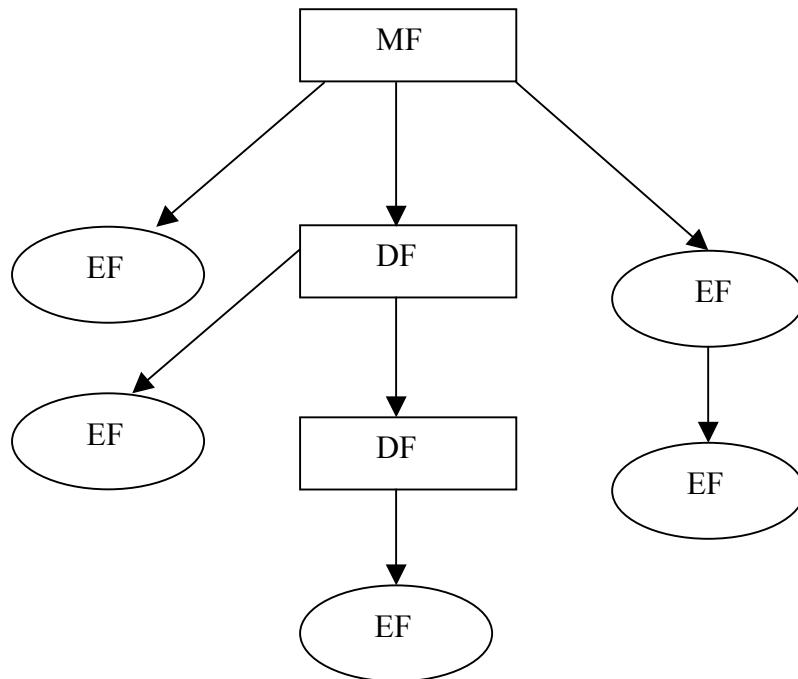
APDU benutzt das Protokoll TPDU (transmission Protocol data units) . Es gibt 2 Type für TPDU:

- 1 T=0 Byteorientiert , nämlich das kleinste transportierte und ausgeführte Unit ist Byte .
- 3 T=1 Blockorientiert, nämlich das kleinste tr transportierte und ausgeführte Unit ist ein Block .

2.5 Smart Card Operationssystem

Smart Card Operationssystem unterstützt eine Menge von Befehlen , die Benutzerapplikation bauen können . Smart Card Operationssystem

Kann sowohl die APDU als auch zusätzliche Funktionen und Extensionen .Meiste Smart Card Operationssystem unterstützt ein modernes File System , das ein hierarchische File System Struktur hat .



Das System unterstützt drei Arten von Files

- 1 MF : Master File .Das Basis vom System .Eine Smart Card hat nur ein MF ,das aus DF und EF besteht .
- 2 DF : Dediziertes File . Ein DF kann ein Paar DF und E besitzen .
- 3 EF : Elementares File :Ein Data File , das vier Type hat .
 1. Transparent File : Eine Sequenz on Daten Byte
 2. Linear konstantes File : Ein Rekord von konstanter Länge
 3. Linear variables File : Ein Rekord von Variable Umfang.
 4. Zyklische Konstante File :Rekord wird gegen die Uhrriichtung eingesteckt .Falls das Rekord voll ist, wird das File in erste Position eingesetzt , d.h das File erhielt das Rekord 1 .

2.6 Smart Card System

Smart Card System ist verteiltes System , das aus zwei Teilen besteht :

- 1 Host System :Das liegt im Hostcomputer .
- 2 Card System :Das liegt in der Karte .

Fast alle Smart Card Software , die vermittelt die Kommunikation zwischen Userapplikation und Card , laufen im Hostcomputer .Die Software unterstützen alle Infrastruktur der Java Card (z.B Sicherheit , Schlüsse Management) .Host Software werden meistens in high-level Programmiersprache geschrieben (z.B Java) .

Card Software sind die in der Smart Card laufenden Software. Card Software besteht auch aus System Software und User Applikation Software . Die Beiden kontrollieren die I/O Kommunikation von Card mit Host ,Sichern die Datenintegration und Sicherheit . Card Applikation besteht aus den Daten und unterstützt die Operation der Daten .Card Software

kann entweder in Assemblesprache oder high-level Programmiersprache implementiert werden , die von dem Mikroprozessoren interpretiert werden kann .

Da die Smart Card von verschiedener Firmen produziert sind , gibt es verschiedenes Smart Card Systeme .Durch die Entwicklung der Java Card Technology , bietet die Java Card Industrie die gesamte Open

Plat-Form und OpenCard Framework ,die in Java Sprache geschrieben werden , die JCRE (Java Card runtime environment) unterstützen .

In den letzten Jahren werden viele Smart Card Standards und Spezifikationen definiert .z.B

- 1 ISO 7816 Standards , die von International Organization for Standardization (ISO) ausgeliefert werden.
- 2 GSM ,die von ETSI (The European Telecommunication Standards Institute für Telefonsystem ausgeliefert werden .
- 3 EMV , die von Europay , MasterCard , und Visa für Finanzielle Industrie ausgeliefert werden .
- 4 Open Platform , die eine integrierte Umgebung für die Entwicklung und Operationen der Multiapplikation-Smart Card System .
- 5 OpenCard Framework (OCF) , die am Anfang von IBM produziert werden .Die eine Applikationsframework des Host , die bietet ein formales Interface um die Kommunikation zwischen Host und Card Leser mit in Card liegende Funktionen zu vermitteln .
- 6 PC/SC (Interoperability Specification for ICCs and Personal Computer Systems)von PC/SC Workgroup definiert wird , die eine universale Architektur definiert , um Smart Card auf dem personalen Computer System zu nutzen .PC/SC und OCF haben Viele ähnliche Konzept .Als OCF auf einem Windows Plattform läuft , kann OCF die Card Akzeptanzgeräte durch den Installierten PC/SC Resource Manager zugreifen .

3. Überblick der Java Card Technologie

Mit der Hilfe der Java Card Technologie kann die in der Smart Card laufenden Programme durch Java Programmierungssprache geschrieben .

3.1 Überblick der Architektur

Tatsächlich ist Smart Card zur Zeit das kleinste Computersystem .Die Kapazität von Speicher der Card ist: 1K für RAM , 16K für EEPROM 24K Für ROM . Wir ziehen nur einen Teil nämlich Subklass von Java Programm , um die Java Visual Maschine zu implementieren , weil die Kapazität der Speicher sehr geringe ist .

JVM besteht aus zwei teilen :

- 1 Run Off-Card
- 2 Run On-Card

Manche nicht in der Ausführungszeit laufende Prozess können in der Off-Card System durchgeführt .Java Card Technologie definiert eine

JCRE (Java Card Runtime Enviroment) , die den Card Speicher , Kommunikation .Sicherheit , Applikationsausführung Model unterstützt . Java Card Technologie definiert ein Platform , das aus Drei Spezifikationen besteht :

- 1 Java Card 2.1 Visual Maschine (JCVM) Spezifikation : ein Submenge von Java Sprache und Visual Maschine .
- 2 Java Card 2.1 Runtime Environment (JCRE) Spezifikation , die Beschreibe Java Card Runtime Verhältnis , Speicher Management , Appletmanagement und andere Runtime Charakteristiken .
- 3 Java Card 2.1 Application Programming Interface (API) Spezifikationen .Die beschreibt eine Menge von Kern und Extension der Java Package und Klasse für die Programmierung der Smart Card Applikationen .

3.2 Java Card Sprachesubmenge

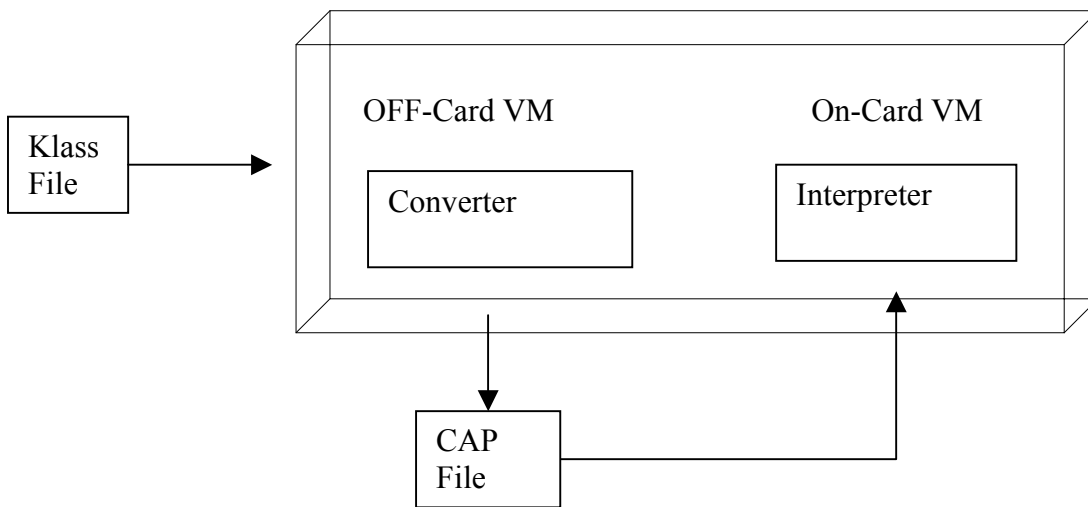
Weil es in Der Smart Card so geringe Speicherplatz gibt , unterstützt die Java Card Technologie nur einen teil von Java Sprache .

Unterstützte Java Charakter	Ununterstützte Java Charakter
1 Klein Primitive Datentyp: Boolean , Byte , Short	1 Datentyp : Long , double Float
2 Eindimensionale Array	2 Charakter und String
3 Java Packages , Klasse ,	3 mehrdimensionale Array
4 Die int Keyword	4 Dynamike Klassaufladen
	5 Sicherheit Manager
	6 Threads
	7 Objekt klonen

3.3 JCVM

Die unterschiede zwischen JCVM und JVM ist ,dass JCVM aus zwei Teilen besteht :

- 1 Off-Card VM, die den Interpreter hat .
- 2 On-Card VM , die Converter hat .

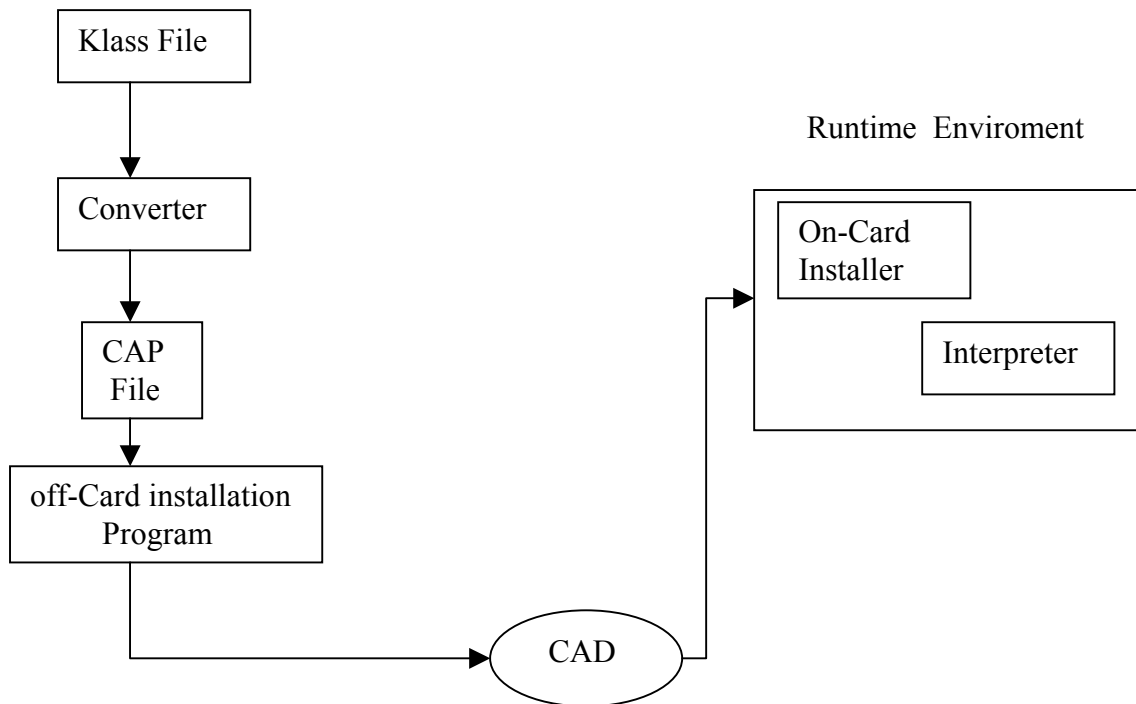


3.4 Java Card Installer und Off_Card Installationsprogramm

Die JCAM implementiert alle Funktionen der Virtual Maschine : laden Java Klasse File auf und konvertiert zum CAP File , die von Interpreter durchgeführt wird .Ein CAP File ist ein binäres Formatfile , das aus einer Menge von Komponenten (z.B Klasseinformation , ausführbare Bytecode , Linkinformation und Verifikationsinformation) besteht . Zusätzlich bei der Produktion des CAP File erzeugt das Converter ein Export File , das Public APIs von das Package beschreibt .Ein Export File kann nicht implementiert werden .Es hat nur Verifikation- und Linkinformation und kann als the Kopf von C Programmiersprache betrachtet werden .

Das Converter hat die Aufgabe , die Aufgeladenen Files in ein Package und dann ein CAP zu konvertieren .Bei der Produktion erzeugt das Converter auch ein Export File ,um die Anfertigung zu erkundigen .

Die Aufgabe von Interpreter ist , die CAP File bzw. Bytecode auszuführen und Speicherverteilung zu kontrollieren und die Runtime Sicherheit . Die Interpreter kann die CAP Files nicht aufladen . Die überlässt die Java Card Technologie einen in der Card liegenden Unit Installer .Der Installer arbeitet mit einem Off-Card Installation Programm ,das das CAP File zu die JVRC durch eine Card Akzeptanzgeräte überträgt .Der Installer schreibt binäre Kode in den Codespeicher , markiert es mit den anderen in der Card schon existierenden Klassen , erzeugt und initiiert alle schon von JVRC benutzten Datenstrukturen .



3.5 JCRE

JCRE (The Java Card Runtime Enviroment) besteht aus Komponenten das Java Card Systems . Die ist zuständig für Speicher Management der Card , Netzwerkkommunikation , Ausführung und Sicherheit das Applet .Man kann JCRE als das Betriebssystem der Smart Card betrachten . Wie folgende gezeigt , besteht JCRE aus VIE Teilen :

- 1 JCVM : Der Interpreter des Bytekode .Der führt Bytekode aus , kontrolliert die Verteilung des Speichers und die Objekte ,verbessert die Sicherheit .
- 2 APIs : Java Card Applikation Framework Klasse . Die definiert den Interface der Applikationsprogrammierung .Mit der Hilfe von APIs kann man den Applet einigermaßen leicht erzeugen , so dass sich der Programmierer nicht um die Infrastruktur der Applet kümmern sondern auf die Auswirkung der Programme konzentrieren
- 3 Industry Specific Extension : Um die Forderung der Spezialen Industrie gibt es noch umfangreicher APIs , die viele zusätzliche Funktionalitäten besitzt .
- 4 JCRE Systemklasse : Die System Klasse tun wie JCRE ,also wie den Kern ein Operationssystem .Die kontrollieren die Transaktion und Kommunikation zwischen den Host und Java Card Applet .Die ist für die Erzeugung , Selektierung und Unselektion des Applet zuständig .

Das Lebenszeit von JCRE ist eben so lang wie das Lebenszeit von Card ,d.h. wenn das Applet in der Card aufgeladen ,aktiviert die JCRE auch . Die JCRE initiiert VM und erzeugt Objekt , um die JCRE Services anzubieten und Applet zu kontrollieren . Eine Außergewöhnliche Eigenschaft von JCRE ist ,dass die abgebrochene Prozedur bzw. Transaktion von Anfang wiederhergestellt wird , weil der Status der Information der Card in EEPROM gespeichert ist . Nachdem die Prozedur wieder aufgerufen wird , ist die in EEPROM liegenden Information wieder zur Verfügung gestellt .Nur wenn die Prozedur richtig total durchgeführt ist, ist die in EEPROM gespeicherten Informationen verändert .

Eine CAD Session ist die Zeit , wobei die Card in die CAD eingesteckt wird und die JCRE läuft ,bis die Card ausgezogen ist .Während einer CAD Session macht die JCRE genau wie eine Smart Card , die APDU I/O Kommunikation mit einer Host Applikation unterstützt . APDUs (Application Protocol Data Unit) sind Daten Packets , die zwischen Applets und die Host Applikation umgetauscht .

Die JCRE unterstützt zusätzliche drei Runtime Eigenschaft.

- Persistent and transient Objects
- Atomic Operations and Transactions
- Applet Firewall and die Sharing mechanismus

3.6 Java Card APIs

Die Java Card Apis besteht aus einer Menge der Klasse für Card Applikations . Die APIs besteht aus drei Kernpackages und einer Extensionspackage .

1. Java. lang Paket : Die ist eine Subklasse von Java.lang Paket auf dem Java Platform .Die unterstützten Klasse sind Objekte , Throwable, und einige VM Exceptionklasse .

Object	Throwable	Exception
RuntimeException	ArithmeticException	ArrayIndexOutOfBoundsException
ArrayStoreException	ClassCastException	IndexOutOfBoundsException
NullPointerException	SecurityException	NegativeArraySizeException

2. Javacard.framework Package :Die Javacard.framework ist ein echter Package, der Frameworkklasse und Interface für die Kernfunktion der Java Card Applet anbietet .Die Java Platformklasse java.lang.System ist nicht vorhanden .

3. Javacard.security Package :Der bietet eine Framework für die kryptographische Funktionen von der Java Card Platform .

4. Javacardx.crypto Package :Der ist ein Extensionspackage , der die abstrakte Basisklasse Cipher ,um die Enkryptions- und Dekryptionsfunktion definiert .

3.7 Java Applet Entwicklungsprozess

Wie das folgende Bild gezeigt beginnt der Durchlauf einer Java Card Applet .

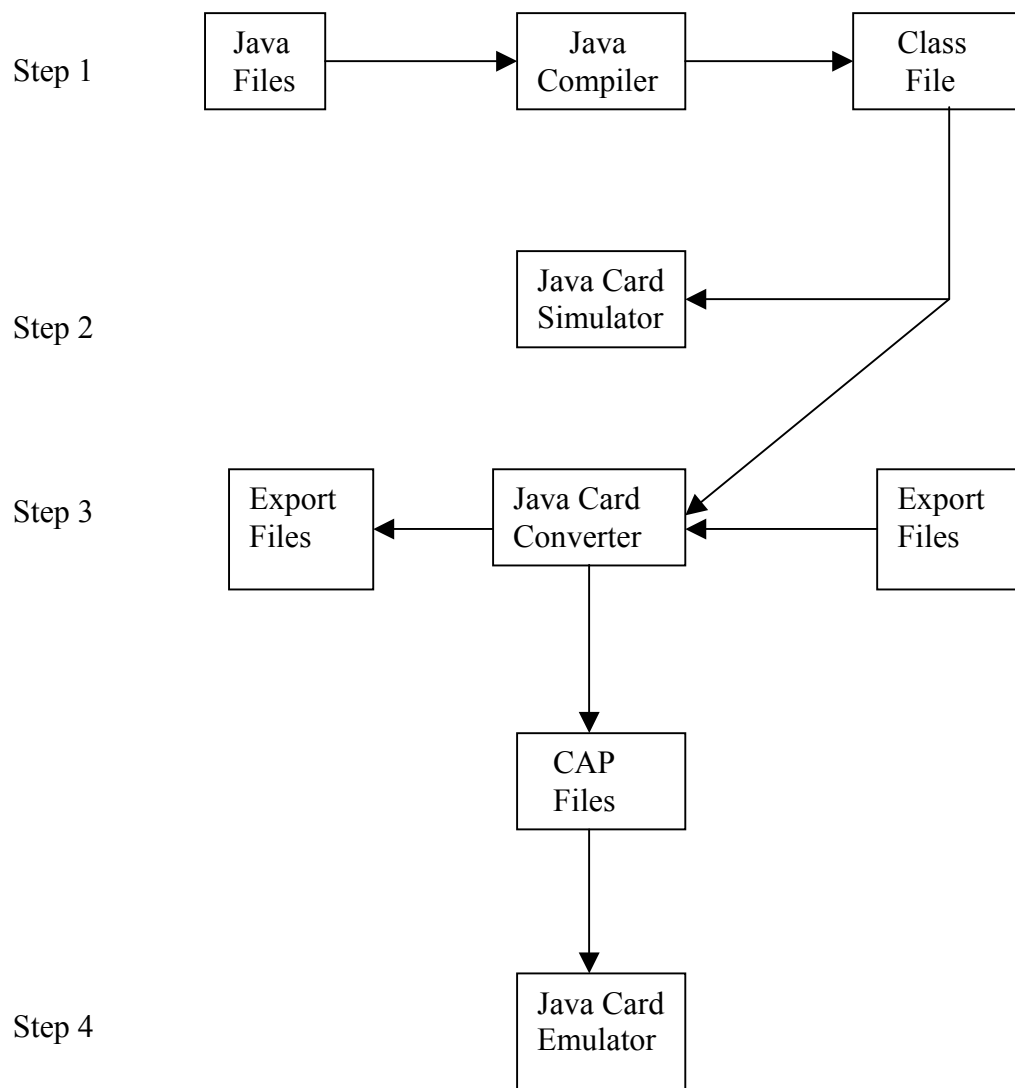
1. Eine oder mehrere Java Klasse wird in die Sourcecode übersetzt und einige Klasse Files werden erzeugt .

2. Die Applet läuft und wird in einer Simulationsumgebung getestet . Die einige Eigenschaften (z.B AppletFirewall und Transient und persistent Verhältnis of Objekte) können nicht getestet .

3. Die von Klassefiles kombinierten Java Pakete werden zu einer CAP File mit der Hilfe des Java Card Konverter Konvertiert . Bei der Konvertierung gibt es noch einen oder mehrere Export Files . Falls die Applets Paket konvertiert wird , kann der Konverter auch einen Export File erzeugen.

4. Die Cap File wird in eine Emulationsumgebung Aufgeladen und getestet. In dieser Phase wird nicht nur Applet weiter getestet, sondern auch die Runtime Verhältnis von der Applet gemesst .

Nach der Emulation ist die Applet bereits zu installieren.



3.8 Applet Installation

Falls eine Java Smart Card ist anfertigt , einige notwendige System and die Java Card Runtimeumgebung werden ins ROM gespeichert . Solcher Prozessor heißt Masking .Nach der Anfertigung der Java Card können die Java Card Applet in die Card aufgeladen werden .Es muss bemerkt werden ,dass die aufgeladenen Klasse die Subklasse von der in der Card gespeicherten Klasse .

4 Java Card Objekte

In der Java Card Technologie erzeugt die JCRE and Applet Objekte , um die Daten zu repräsentieren , speichern und manage . Die lauffähigen Applets sind Objekte von der AppletKlasse .Die allen Objekte haben den gleichen Rootklasse `java.lang.object` .Die Werte in einem neuen Objekt oder Komponente in einem neuen Array wird als zero, null oder false definiert .

Java Card Technologie unterstützt beide persistente and transiente Objekte .

4.1 Java Card Memory Model

Es gibt drei Memory Model von Java Card :

- ROM :nur lesbar und ist teuerst von die drei Model .Programm und Daten werden bei Anfertigung der Card in ROM gespeichert , und nie verändert .
- RAM :Lesbar und schreibbar .Die Daten werden gelöscht , falls der Strom ausgefallen ist .Ein RAM Zell ist viel Mal grösser wie den Zell von EEPROM .
- EEPROM :Lesbar und schreibbar . Die Daten werden nicht gelöscht , sogar wenn der Strom ausgefallen ist .Außerdem ist Geschwindigkeit des Schreiben von EEPROM viertel Schnell wie ROM .

4.2 Persistent Objekte

Das Speicher und Daten der persistente Objekte sind durch CAD Session gespeichert .Es gibt einige Eigenschaft für persistente Objekte :

- Ein persistenter Objekt wird durch `new` Operator erzeugt .
- Ein persistenter Objekt hat Status und Werte während die CAD Session .
- Irgendein Update kann nur realisiert werden , sofern die Prozedur erfolgreich durchgeführt wird . Sonst bleibt die Werte unverändert .
- Ein persistent Objekt kann von ein transienter Objekt aufgerufen .
- Ein transienter Objekt kann auch den transienten Objekt aufrufen.
- Falls persistent Objekt kann nicht von andere transient Objekt aufgerufen , dann wird der unerreichbar und vernichtet .

4.3 Transiente Objekte

Der Inhalt eines transiente Objekt hat ein vorläufige Eigenschaft . Im Vergleich zum persistente Objekt wird das Speicher für transintem Objekt reserviert und kann nicht vernicht , sofern der Garage nicht implementiert wird .

Ein Applet soll bei seinem Lebensdauer nur einen transitorischen Objekt erzeugen und den in einen persistenten Feld speichern. Als der Strom eingeschaltet wird , benutzt der Applet die gleiche Reverenz , die transient Objekt zu greifen . Jedoch ist die Objektdaten von der vorigen Session schon verloren.

4.3.1 Eigenschaft der transient Objekte

Ein transienter Objekt im Java Card Platform hat die folgenden Eigenschaften:

- Ein transitorischer Objekt wird durch Aufruf der Java Card APIs erzeugt .
- Ein transient Objekt hat nicht Status und Werte durch die CAD Session .

- Falls der Strom ausgefallen ist , oder ein Fehler während dem Update erscheint , wird die Werte des transienter Objekts überschrieben .
- Ein transienter Objekte kann von einem persistenter Objekt aufgerufen .
- Ein transienter Objekt kann einen persistenter Objekt aufrufen.
- Wenn ein transienter Objekt nicht von anderem Objekt aufrufen kann , ist der unerreichbar und wird von Garbage weggenommen .

Die Eigenschaften von transienter Objekte ermöglicht , ideal für Small amounts der vorläufige Applet Daten , die nicht durch die CAD Session reserviert werden .

4.3.2 Transienter Objekte Type

Es gibt zwei Arte von transienten Objekte , die verursacht ,dass die JCRE den Feld der Objekte vernichtet , falls die entsprechenden Event erscheint .

- CLEAR_ON_RESET :wird für Datenspeichern benutzt , der durch die Applet Selektion und nicht durch Card Reset hinlegt .
- CLEAR_ON_DESELECT. Wird für die Objekte benutzt, der speichert werden muss , solange ein Applet ausgewählt wird,und nicht durch die Applet Selektion und oder Resets .

4.3.3 Erzeugung der transienten Objekte

In der Java Card Technologie wird transienter Objekte durch die Benutzung einer der Methode von JCSYSTEM klasse .

Methods	Result of the method call
Public static boolean[] makeTransientBooleanArray(short Length, byte event)	Create a transient boolean array
Public static byte[] makeTransientByteArray(short length, Byte event)	Create a transient byte array
Public static ahort[] MakeTransientShortArray (short length , Byte event)	Create a transient short array
Public static Objekt[] MakeTransientObjektArray (short length , Byte event)	Create a transient Object array

Die JCSYSTEM bietet eine einfache Fragmethode an, so dass die Applet entscheidet , ob ein zugriffener Objekt transient ist .

Public static byte isTransient (Object theObject)

Weil de Speicher in einer Smart Card ziemlich gering ist , kann man die beiden Objekte nicht grenzfrei erzeugen .Falls der Speicher in RAM nicht ausreichend , liefert die JCRE eine SystemException aus.

Sofern die Objekte erzeugt werden , sind die immer erreichbar , solange die von dem Stack aufrufbar sind. Falls alle Links eines Objekts unerreichbar , ist der von Garbage weggenommen .

5 . Atomicity und Transaktionen

In dem Bereich der mobilen und verteilten Umgebung zur Speicherung der privaten Daten wird Smart Card als eine vorgezogene Geräte betrachtet .Jedoch gibt es eine Risiko bei der Durchführung eines Applets .Z.B es gibt ein Fehler bei der Berechnung ,der Strom ist plötzlich ausgeschaltet . Solche vorzeitige Aktionen von Smart Card der CAD wird als genannt .JCRE bietet ein robustes Mechanismus an , um die atomike Operation zu sichern .Das Mechanismus besteht aus zwei Stufen :

- Java Card Platform garantiert , dass jede Veränderung im Feld des persistenten Objekt oder ein Klassefeld atomik ist.
- Java Card Plastform unterstützt einen Transaktionsmaodel

5.1 Atomicity

Atomicity bedeutet , jede Veränderung in einem persistenten Objekt oder einem Klassefeld wird nur durchgeführt , falls die Transaktion komplett und fehlerlos beendet .Sonst bleibt die betroffenen Daten unverändert .Es gibt doch noch eine Ausnahme .Für das transiente Array unterstützt JCRE die Eigenschaft von Atomicity nicht .

5.2 Block Data Update in einem Array

Die klasse javacar.framework.util bietet eine Methode arrayCopy , die Atomicity für Block Data Update von Multiple Daten in einem Array sichern.

```
Public static short arrayCopy
```

```
(byte[] src , short srcOff , byte [] dest , short desOff , sort Length )
```

Die util.ArrayCopy Methode garantiert , die Werte der Elemente von persistenter Objekt seine originale Inhalt wieder speichert wird , falls die Transaktion abgebrochen ist .Jedoch ist diese Funktion für die teansienter Objekte ungültig .Aber benötigt arrayCopy zusätzliche EEPROM , die Atomicity zu unterstützen. , deshalb ist die Geschwindigkeit des Schreiben ziemlich Langsam , weswegen braucht der Apple keine Atomicuity für Array Update .Die Methode util.arrayNonAtomic ist für diesen Zweck .

```
Public Static short arrayCopyNonAtomic
```

```
( byte[] src , short srcOff , Byte[] dest , shirt desOff , short Length )
```

Die Methode Util.arrayFileNonAtomic bietet die Eigenschaft an , dass die Elemente bei der stromlosen Etage nicht automatisch überschreibt wird .

5.3 Transaktion

5.3.1 Commit Transaktion

Eine Transaktion beginnt mit dem Aufruf der Methode JCSYSTEM.beginTransaction und beendet mit dem Aufruf der Methode JCSYSTEM.commitTransaktion .Die Veränderung in einer Transaktion ist konditional , sofern die Methode JCSYSTEM.commit durchgeführt ist .

```
//begin a transaction
```

```
Jcsystem.beginTransaction()
```

```
//all modifications in a set of persistent data
```

```
//are temporary until the transaction is committed
```

```
...
```

```
// commit a transaction
JcSystem.commitTransaction
```

5.3.2 Stoppen Transaktion

Transaktion kann entweder durch einem Applet oder JCRE gestoppt . Falls irgendein internerer Fehler existiert , wird die Transaktion von der Methode `JCSystem.abortTransaction` gestoppt .Die originale Werte werden wieder in die ursprünglichen Stelle gespeichert . Außerdem muss die Transaktion im lauf sein ,sonst liefert JCRE eine `TransactionException` .Auf jeden Fällen wird die Transaktion von `JCSystem.abortTransaction` beendet ,falls irgendein ungewöhnliches Ereignis erscheint .

5.3.3 Vernetzte Transaktion

Während dem Lauf darf nur eine Transaktion durchgeführt werden ,weil die Smart Card sehr geringe Arbeitsplatz zu Verfügung hat .Die Methode `JCAyatem.transactionDepth` ist dafür zuständig , ob ein Transaktionen im Lauf gerade liegt .Die liefert 1 für ja , 0 für nein .

5.3.4 Commit Kapazität

JCRE bietet ein commit buffer an , um die originalen Inhalte der Objekte zu speichern .Falls die Transaktion ungewöhnlich beendet ist ,können die Objekte ihre originalen Werte erhalten .Das hat im heutigen Industriebereich eine sehr große Bedeutung .Die Kapazität der Commit Buffer ist meistens ausreichend , um die Transaktion zu führen .Es ist empfehlenswert , dass nur die notwendigen Informationen in den Commit Buffer hingebacht werden .Die folgenden zwei Methoden kann den Applet helfen , wie viele Commit Kapazität in einer Java Card Platform verfügbar ist:

- `JCSystem.getMaxCommitCapacity()` :zeigt die gesamte Byteanzahl im Commit Buffer .
- `JCSystem.getUnusedCommitCapacity()` :zeigt die gesamte verfügbare Byteanzahl im Commit Buffer.

Falls die Commit Kapazität in einer Transaktion nicht ausreichend ist ,liefert JCRE eine Meldung `TransactionException` aus .

5.3.5 TransaktionException

`TransactionException` ist eine Subklasse von `RuntimeException` .Die Folgenden sind die Klasse `TransactionException` :

- `IN_PROGRESS-beginTransaction` :aufgerufen , falls eine Transaktion bereits im Lauf liegt .
- `NOT_IN_PROGRESS_commitTransaction` oder `abortTransaction` :aufgerufen , falls eine Transaktion nicht im Lauf liegt .
- `BUFFER_FULL`: falls Commit Buffer voll ist .
- `INTERNAL_FAILURE` : ein interneres Problem existiert im Transaktionssystem .

5.3.6 Lokale Variable und transiente Objekte während einer Transaktion

Wir müssen klar sein, dass in einer Transaktion nur persistente Objekte teilnehmen. Für die Lokalen Variable und transiente Objekte haben nicht damit zu tun. Die folgenden zeigt, wie die drei Copy Operationen bei Aufruf einer transiente Array `key_buffer`. Falls die Transaktion ungewöhnlich endet, werden die Elemente von Arraycopy Operation jedes einzelne Update nicht von JCRE geschützt und die Variable bekommt die Werte 1.

```
//byte[] key_buffer =JCSytem.makeTransientByteArray
                        (KEY_LENGTH ,JCSytem.CLEAR_ON_RESET );
JCSytem.beginTransaction()

Util.arrayCopy(src , src_off , key_buffer,0 ,KEY_LENGTH );
Util.arrayCpyNonAtomic (src_off m,Key_Buffer,0,KEY_ENGTH);

For (byte I=0; I <KEY_LENGTH ;i++)
    Key_buffer[i] = 0 ;

Byte a_local = 1;

JCSytem.abortTransaction();
```

Literatur

[1] Lit.: Chen: JavaCard Technology for SmartCards. Addison-Wesley 2000.

[2] Lit.: Bruce Eckel: Thinking in Java. 04.1999