le cnam

Java Card Technology

Samia Bouzefrane

Associate Professor

CEDRIC –CNAM

samia.bouzefrane@cnam.fr
http://cedric.cnam.fr/~bouzefra

le cnam

Java Card technology: introduction and principles

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card - Introduction

- **Need to « programmable » systems**
- **Need to « evolutive » solution (exceed the ROM)**
- **Applications :**
  - ✓ Long to develop
- **Attemps**

1st version: october 1996, startup and actual product in 1998, an industrial reality since 2000. In 2004, the number Java Cards sold has reached one billion.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Stages of industry development

➢**The smart card and the main stages of development technology:**

✓ The pioneers (1975-1985): first thoughts
(the technological basis established)

✓1985-1995: the technology is improved
- Markets and large deployments: CB, GSM
- Limits: need more flexibility

✓1995-2005 : explosion of the market, with new paradigm
- cards based on Scalable Java Card

✓2006: 1.2 billion mobile phones using SIM cards / Java Card
1.65 billion smart cards / Java Card (Sun source site)

✓2008: 90% of SIM cards are Java Card in Europe, America.
6 billion Java Card (According to Sun)

✓2005-???: the card becomes an element of the network
- SCWS (Smart Card Web Server)
- .Net, Java Card 3.0

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

The beginning of Java Card technology

➢**November 1996, the first proposed use of Java for cards is made by a team of Schlumberger (Austin)**

- ✓Java Card API proposal for programming in Java Card
- ✓ Java Card 1.0

➢ **Bull, Gemplus and Schlumberger create the Java Card Forum**

- ✓ the JCF discusses and proposes specifications to Oracle/Sun

➢**November 1997, publication of the Java Card 2.0**

Gemplus demonstrates in October / November CASCADE, the first chip 32-bit RISC (ARM 7) with flash memory, "an" implementation of the Java Card 2.0 and DMIs (Direct Method Invocation), etc.

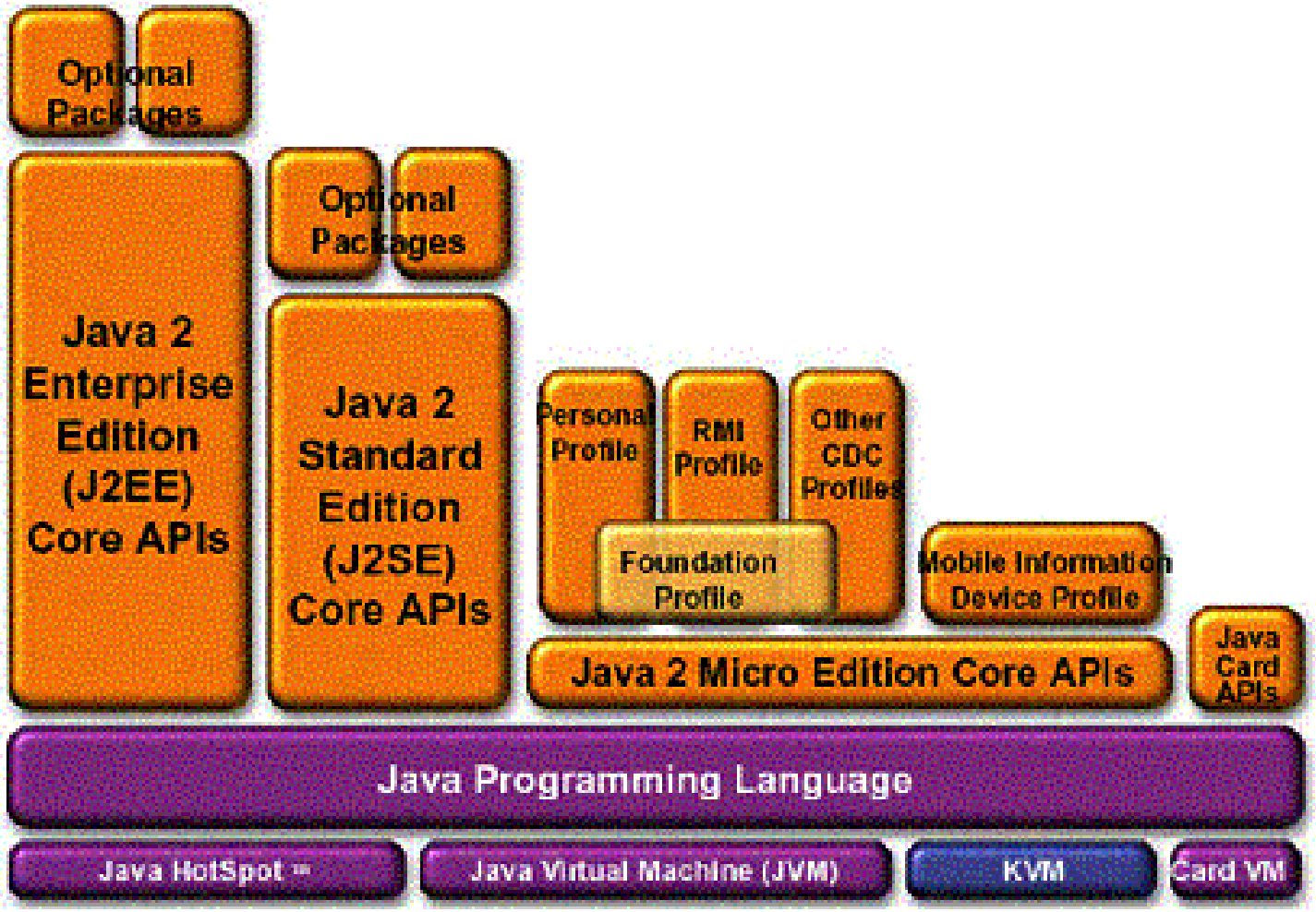
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Evolution to Java Card 2.x

- **The version 2.0 of Java Card Specification :**
  - ✓ a runtime environment
  - ✓The ability to write applets with an object-oriented approach (although the loading format was not specified)
- **March 1999, version 2.1 that includes 3 parts:**
  - ✓ Java Card API Specification
  - ✓Java Card Runtime Environment Specification
  - ✓Java Card Virtual Machine Specification

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

An element of Java technology



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

About the license model / 1

- **The specification is available at:**
  - ✓ http://java.sun.com/products/javacard/
- **Sell cards (with or without logo) and display compatibility with technology means being licensed Java Card Technology**
- **Which provides access to :**
  - ✓A reference implementation
  - ✓Following compatibility testing
  - ✓Specific support

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

About the license model / 2

➢ **Java Authorized Licensees of Java Card Technology**

✓ the companies listed below licensed Java Card technology from the Sun MicroSystems. Only Java Card licensees can ship products that bear the « Java Powered » logo and claim compatibility with the Java Card Platform specification and Java Card TCK.

✓ ARM, Aspects, CCL/ITRL, Fujitsu, Gemplus, SAGEM, Oberthur Card Systems, Trusted Logic, etc.

Source : http://java.sun.com/products/javacard/licensees.html

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card Forum

- **Association of manufacturers of silicon, embedders and customers**
  - ✓ Promote Java Card technology
  - ✓Set of technology choices and then offer it the Oracle "Standard".
- JCF : http://www.javacardforum.org

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

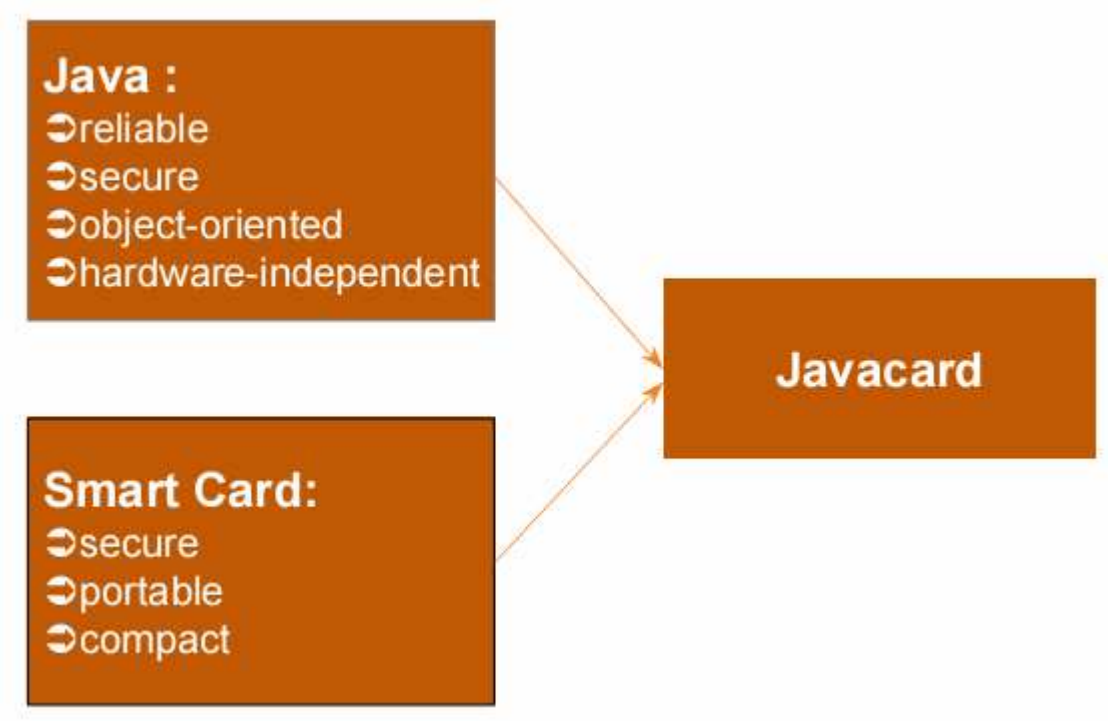
A Java Card platform

- is a smart card
- with a virtual machine
- able to execute applications written in Java

- Java Card platforms are standardized by Oracle and Java Card Forum
- Java is the programming language the most used in the application developement dedicated to smart cards

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card = Java + smart Card



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

A standard smart card

- Application, OS and hardware linked together
- The application is developed only by the owner of the OS
- The application is developed in a low-level language (C, Assembler)
- Development cycle = 5 months
- No true multi-application (data only)

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

A Java Card platform

- Application, OS and hardware are independent
- The application is developed by any Java programmer
- The application is developed in a standard language (high level)
- Development cycle = 2 months
- Multi-application card (code + data)

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card technology advantages

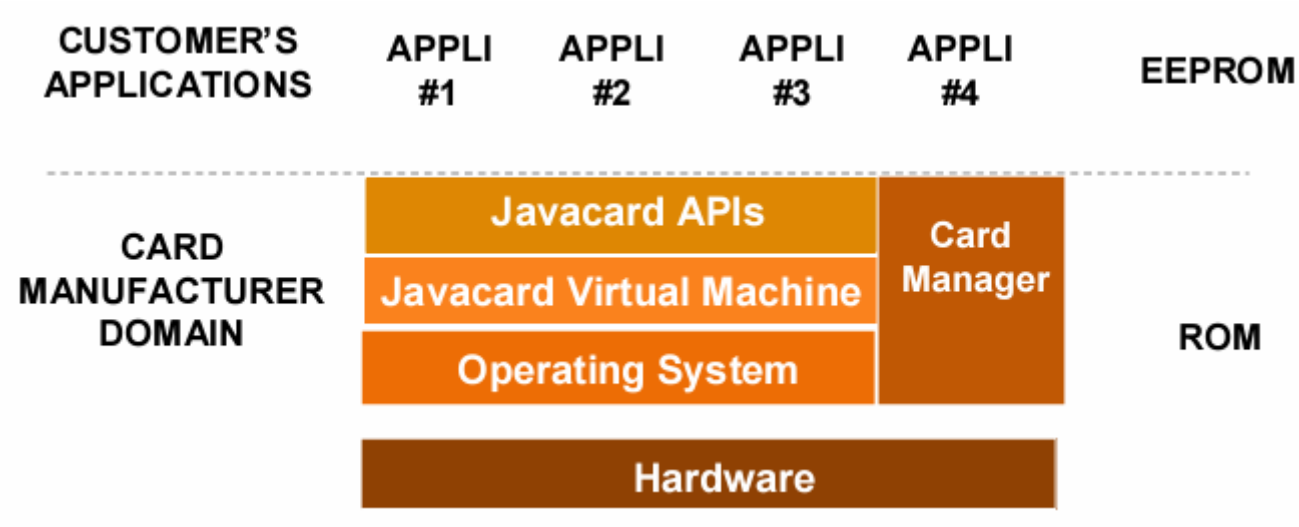
- easy development
- Interoperability of applets (for use on different platforms)
- Safety (of language, optimization, etc.).
- Multi-application
- dynamicity
- Openness and compatibility (addition and update applications)
- Ability to post-personalization

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card language

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card actors



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card characteristics

➢**Card architectures with very small sizes:**

- less than 1K of RAM, 24-28 KB of ROM and 8 to 16 KB NVM (EEPROM).

➢ **To integrate Java technology into a card, the choices are:**

- Reduce language features
- Minimum required to run a Java Card program are:
  - -24 KB of ROM, EEPROM and 16 KB of 1 KB of RAM.
- Distribute the model of the JVM between “on Card” and “off Card “

➢**Three parts :**

✓Java Card API Specification

✓Java Card Runtime Environment Specification

✓Java Card Virtual Machine Specification

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Supported Types

Type	Size	Supported in Javacard
boolean	-	yes
byte	8-bit	yes
short	16-bit	yes
int	32-bit	yes (optional)
long	64-bit	no
double	64-bit	no
char	16-bit	no
String	variable	no
float	32-bit	no

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Not supported features

- No Threads
- No dynamic loading
- No Garbage Collector until version 2.2)
- no cloning
- no multi-dimension arrays

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Features

Supported features	Non Supported features
boolean, byte, short	long, double, float, char, String
One-dimension array	Multi-dimension array
Java package, classes, interface and exceptions	Threads, serialization
Extension, abstract method, Overload and object creation (instantiation)	Dynamic loading of classes
« int » is optional	Security manager

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Key words

- **Supported key words**

abstract, boolean, break, byte, case, catch, class, const, continue, default, do, else, extends, false, final,goto null, package, private, protected, public, return, static, super, switch, this, if, implements, import, instanceof, int, interface, new, null, package, private, protected, public, return, short, static, super, switch, this, throw, true, try, void, while.

- **Non supported key-words**

char, double, float, long, native, synchronized, transient, threadsafe, volatile, finalize

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Specific characteristics of Java Card

- Transient objects (APDU, Reset, Select)
- Atomicity
- Sharing
- Exception management on cards
- specific API: Java Card 2.1.x et 2.2
- special methods to install applets, send APDU commands, etc.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Transient Objects

- **Definition :**
  - ✓ objects whose fields are cleared after an event

- **Characteristics**
  - ✓ The value is cleared and not the object itself
  - ✓ located in RAM
  - ✓ used for temporary data frequently changed

- **Events that reset the temporary objects**
  - ✓ Reset, Select, Deselect.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Atomicity / Transaction

- **Definition :**
  - ✓ **a transaction is atomic if all fields are updated or not at all**

- **Characteristics**

  - if a transaction does not end normally (power failure, card removed, etc.), the data are set to their initial values

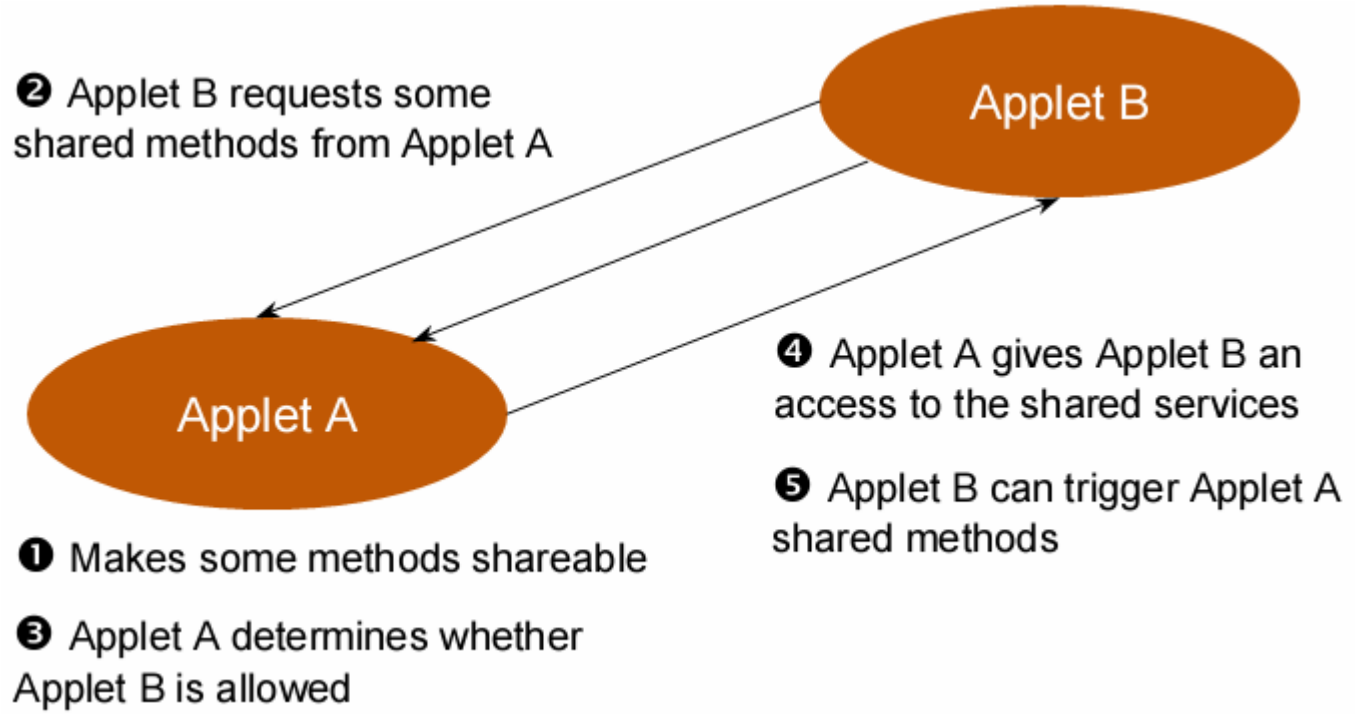
  - prevent the loss of sensitive data (eg. the balance in the wallet)

  - transactional mode can be set or not

  - management of atomicity via the API

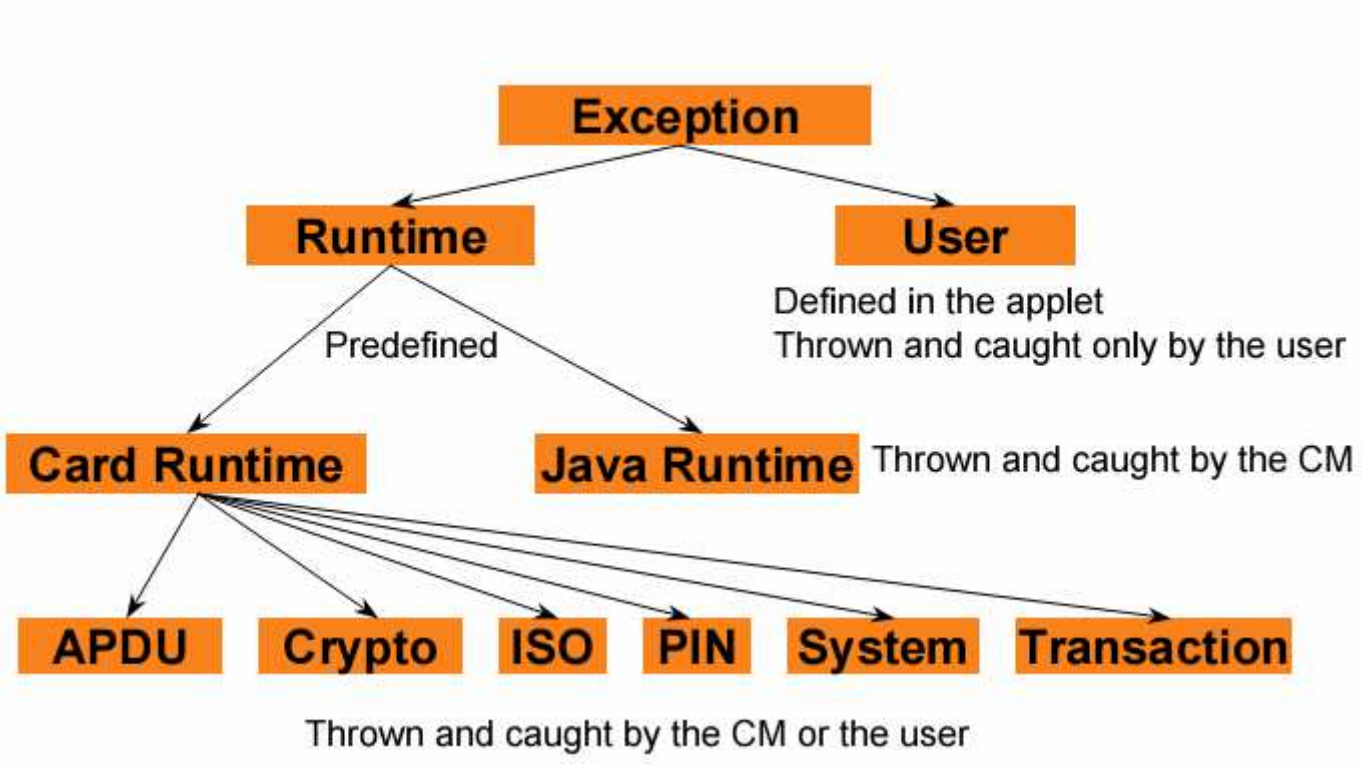
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Sharing



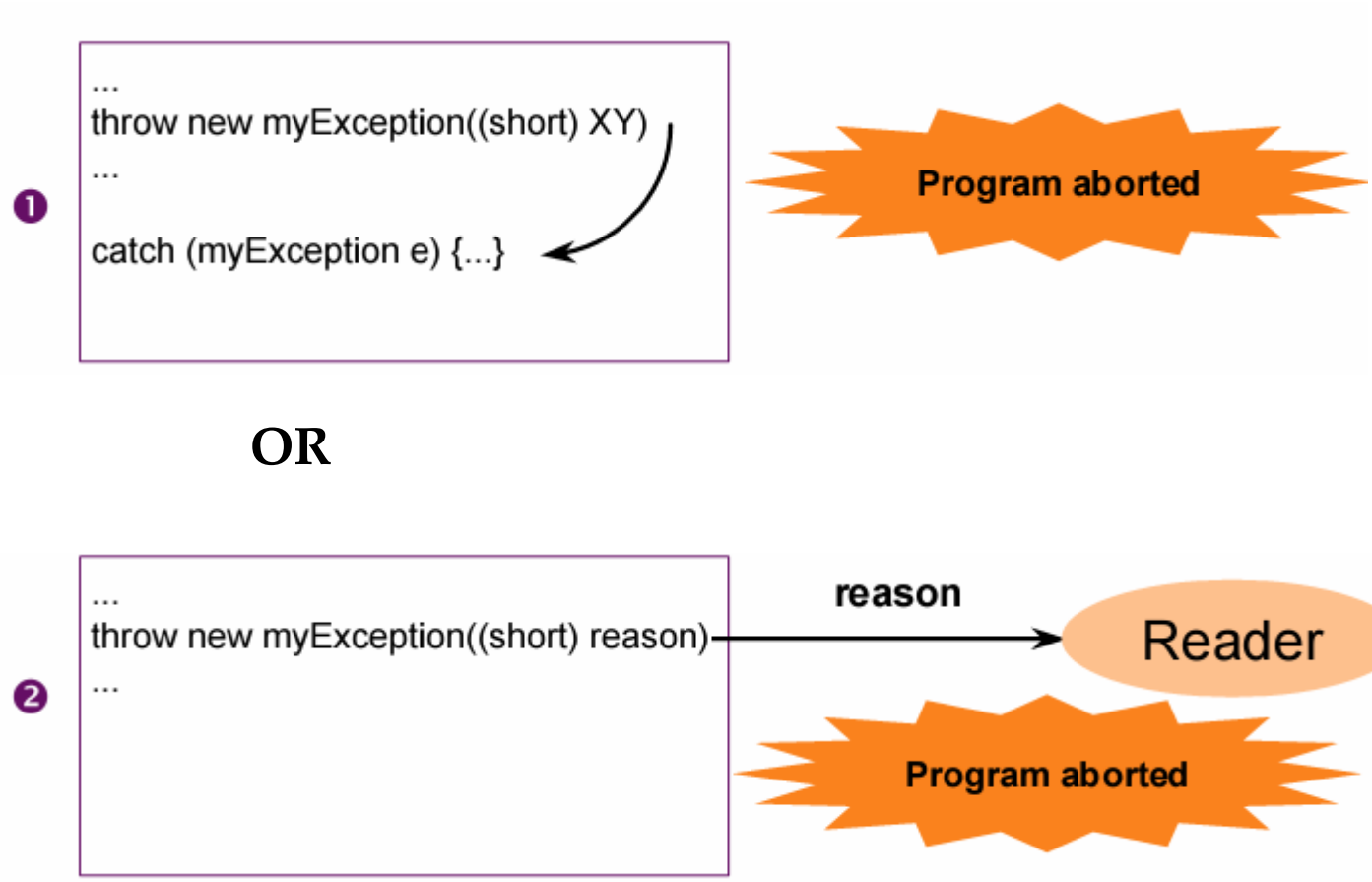
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Card Exception



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Runtime Exception



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exception in Java

- If a method can throw an exception, it must be encapsulated by a `try catch` block.

- Example

```
try
{
   operationWhichThrowsAnException();
}
 catch (Exception e)
   {
         ….
   }
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Exception in Java Card

- `Exception.throwIt(value)`

- Non authorized example

```
if (erreur) throw new ArithmeticException((short)0);
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

Java Card API 2.1

- **3 reference packages**
  - ✓java.lang
  - ✓javacard.framework
  - ✓javacard.security
- **Extension**
  - ✓Javacardx.crypto

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

javacard.framework package

Class `JCSystem`

- **Methods to manage atomicity:**
  - ✓**`beginTransaction`**(): begins transaction
  - ✓ **`commitTransaction`**(): saves data of the transaction into the EEPROM
  - ✓ **`abortTransaction`**(): cancels the transaction

- **Method to manage transient objects**
  - ✓ **`isTransient(Object`**)
  - ✓ **`makeTransientXArray(short`**, byte) X= **`Boolean`**, **`Short`**, **`Object`**

- **Methods to manage sharing**

- **Methods to manage the information system: `getVersion()`**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

javacard.framework Package

- **Contains the card specificities**
- **`Applet` class:**
  - ✓Provides a framework for implementation and interaction with the JCRE
  - ✓Apples must extend this class
- **`APDU` class**
  - ✓For exchanging data with the terminal
- **`PIN` class**
  - ✓ Manages the secret code

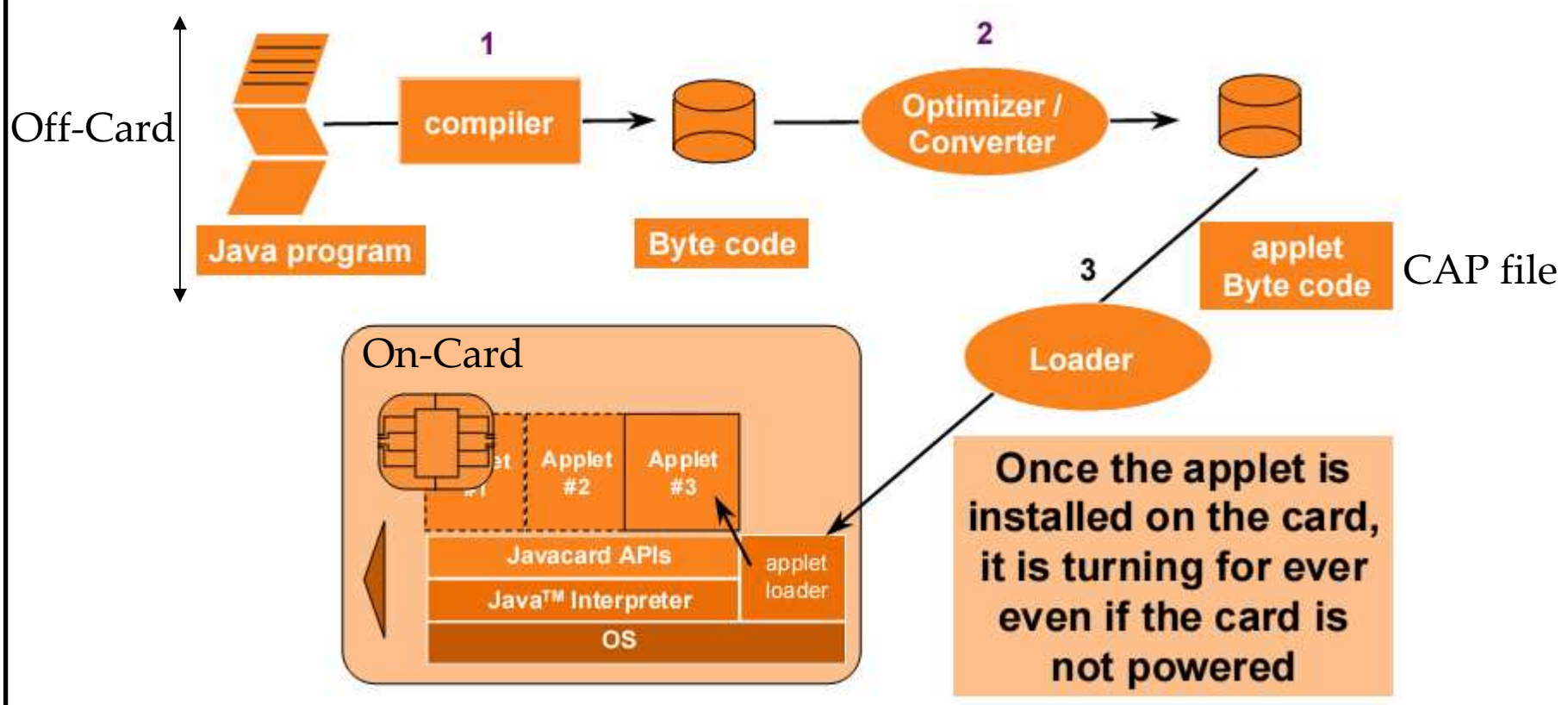
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

javacard.security Package

- Based on `java.security` package
- Allows key management and cryptographic functions
- In addition to the conventional algorithms, it also includes the generation function random number, signature and the calculation of compression functions

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Applet development process

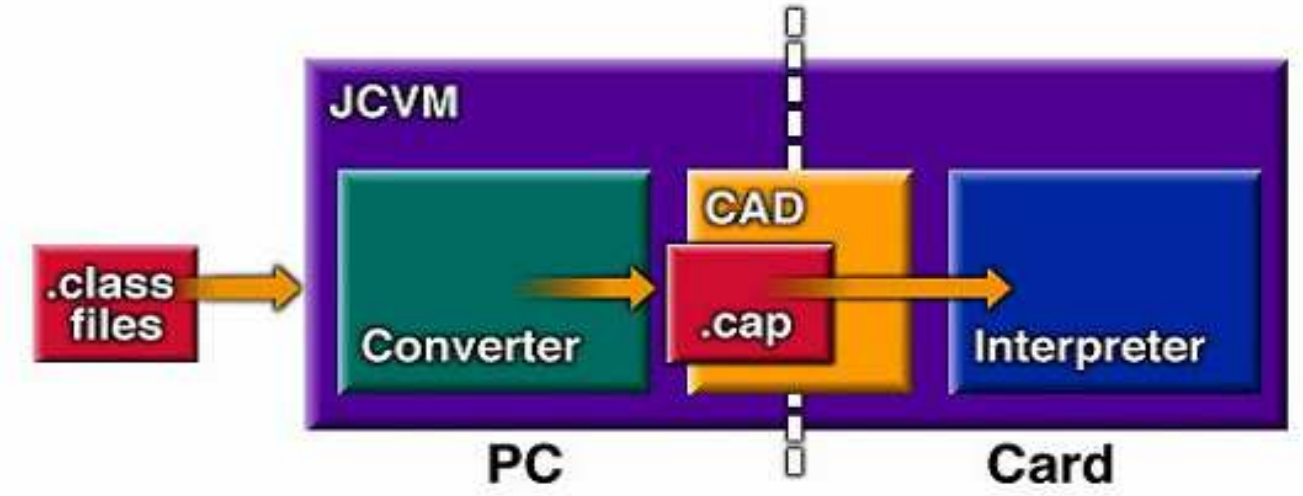


CAP File

- **The « CAP File » contains:**
  - ✓Information on classes
  - ✓ Executable BC (Byte Code)
  - ✓ information necessary to linking
  - ✓Information for verification
- **It has the format of JAR (Java Archive)**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Convertor/Interpretor



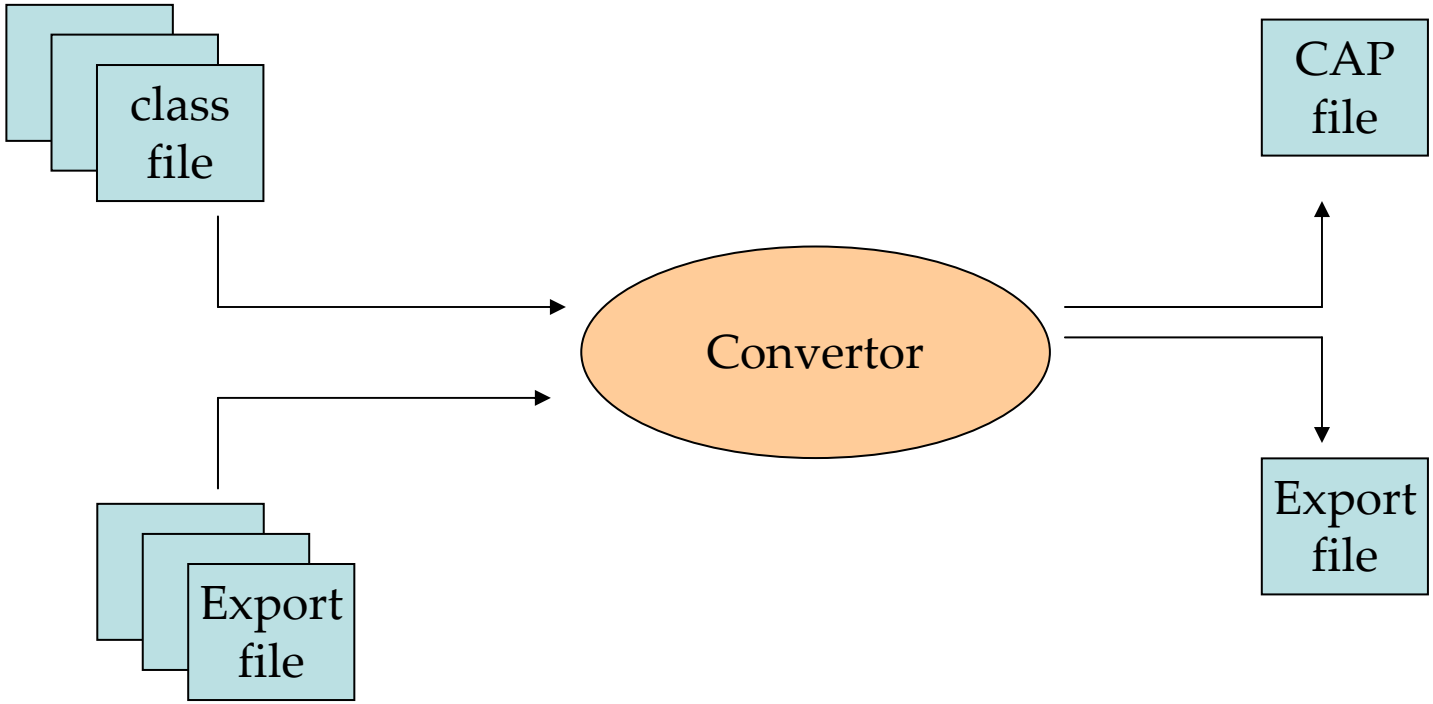
Source: Sebastian Hans, Java Card Platform overview, Sun Microsystems Inc., 2008

Export File

- **The « Export » file is used by the convertor**
- **Information used for linking and verification**
- **Contains information on APIs**
  - ✓Name of the classes
  - ✓Signature of methods
  - ✓Information for linking between packages
- **It does not contain BC, it can be published with an applet allowing the applet Have re-usable objects (shareable)**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Convertor



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Convertor

- **Supports the following operations:**
  - ✓Compliance verification of the Class File Format
  - ✓Testing compliance aspects of the Java language
  - ✓Initialization of static variables
  - ✓Reference resolution (classes, methods and fields) and placed under compact to be more effective in a small system
  - ✓Optimize the byte code
  - ✓Allocation and creation of structures that represent the classes in the JVM

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Interpreter

➢**It provides a runtime environment to run BC of the CAP file. It allows to the applets loaded in a card run to be run on any platform.**

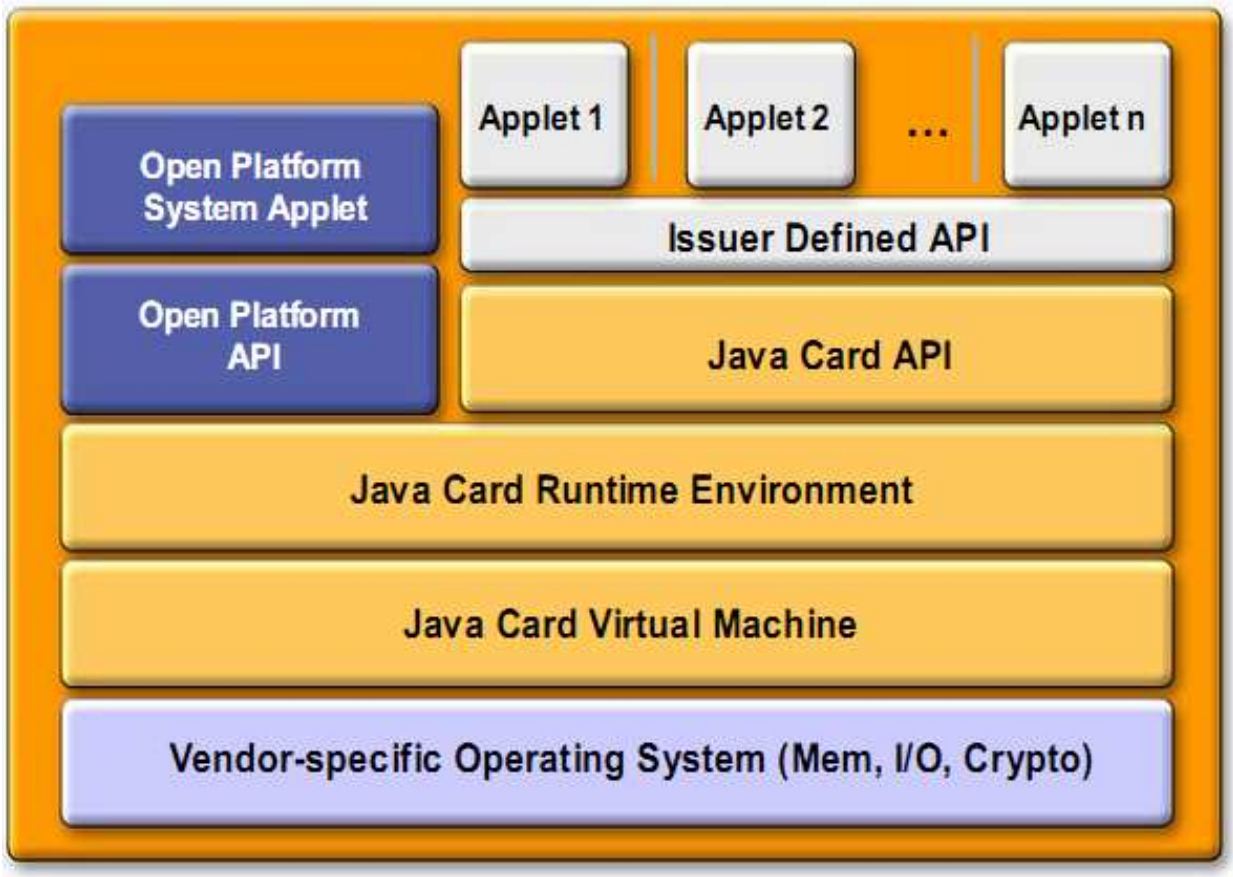
➢ **It performs:**

- ✓The execution of the BC
- ✓The control of the memory allocation
- ✓and ensures safety

➢ **The installation of applets is performed thanks to an applet loader that is distributed between the terminal and the card**

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card Architecture



Source: Sebastian Hans, Java Card Platform overview, Sun Microsystems Inc., 2008

JCRE: life cycle and card session

➢**In workstation environment, the JVM is a process, it is initialized at the begin and then stopped at the end of the process. Objects in RAM are lost.**

➢ **In order that information is retained from one session to another:**

✓In case of a card, the initialization of the JVM is done only once: at the "beginning of life of the card," the objects and data are stored in a non-volatile memory (EEPROM, Flash, etc.).

✓At each session with the card:

- Power: the JCRE is "reactivated"

- The card receives and processes APDU commands

- Turn off: the JCRE is "suspended"

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

JCRE characteristics

- **Persistent objects and temporary**
  - ✓Java Card objects are by default persistent
  - ✓For reasons of efficiency (speed of Read / Write in NVM) and security (key, intermediate results), applets can create temporary objects
- **Atomic operation and transaction**
  - ✓The JCVM ensures atomicity of the updates when modifying object values
  - ✓The JCRE provides an API to allow applets group several rewrites and to provide consistency of these updates (Begin Transaction, Commit, Roll-Back)

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Applet firewall sharing mechanism

- **Each applet runs in its own space**
  - ✓ Applications separated by an applet firewall to prevent intrusion
  - ✓There is a sharing mechanism that allows an applet to access services offered by an applet or by the JCRE.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

How to write an applet ?

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Building Java Card applets

- **An application dedicated to a card**
  - ✓Code in the card: server application = Java Card Applet
  - ✓Code in the terminal: client application
- **An application built in 3 steps**
  - ✓ Writing the server application (applet)
  - ✓ Installation of the Java Card applet
  - ✓ Writing the client application

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Writing a Java Card applet

- **Java Card API 2.1**
- **Stages of development of an applet**
  - ✓ Specify the functions of the applet:

    - specify the AIDs (Application Identity) of the applet and package to which the applet belongs

    - write the body of applet

    - compile (.class)

    - convert (.cap)

    - load within the card

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Development phases of an applet

- Specify the functions of the applet
- Assign an AID to the applet and an AIDs to the package of the applet
- Design programs of the applet
- Define the interface between the applet and the terminal

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Applet behaviour

- **Application written in Java Card**
- **Applet on the card**
  - is selected
  - receives messages from the reader
  - processes these messages
  - returns data to the reader
  - is de-selected.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

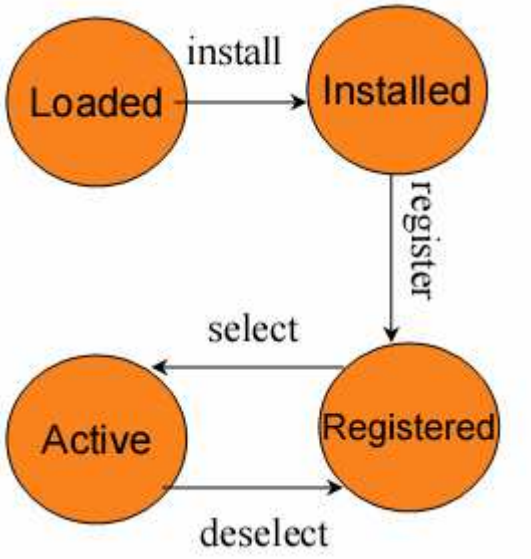
Java Card Runtime Environment

- Loads applets on the card
- Select the applet to activate
- Handles messages (APDUs) received from the reader
- Manages the file system commands and security manager

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Life cycle of an applet

Once the applet is loaded on the card, it must be:
- Installed, registered (identified by the JCRE through its AID)
- Selected (as many applets may be installed on the card)



Specifying the functions of the applet

Example of Echo applet:

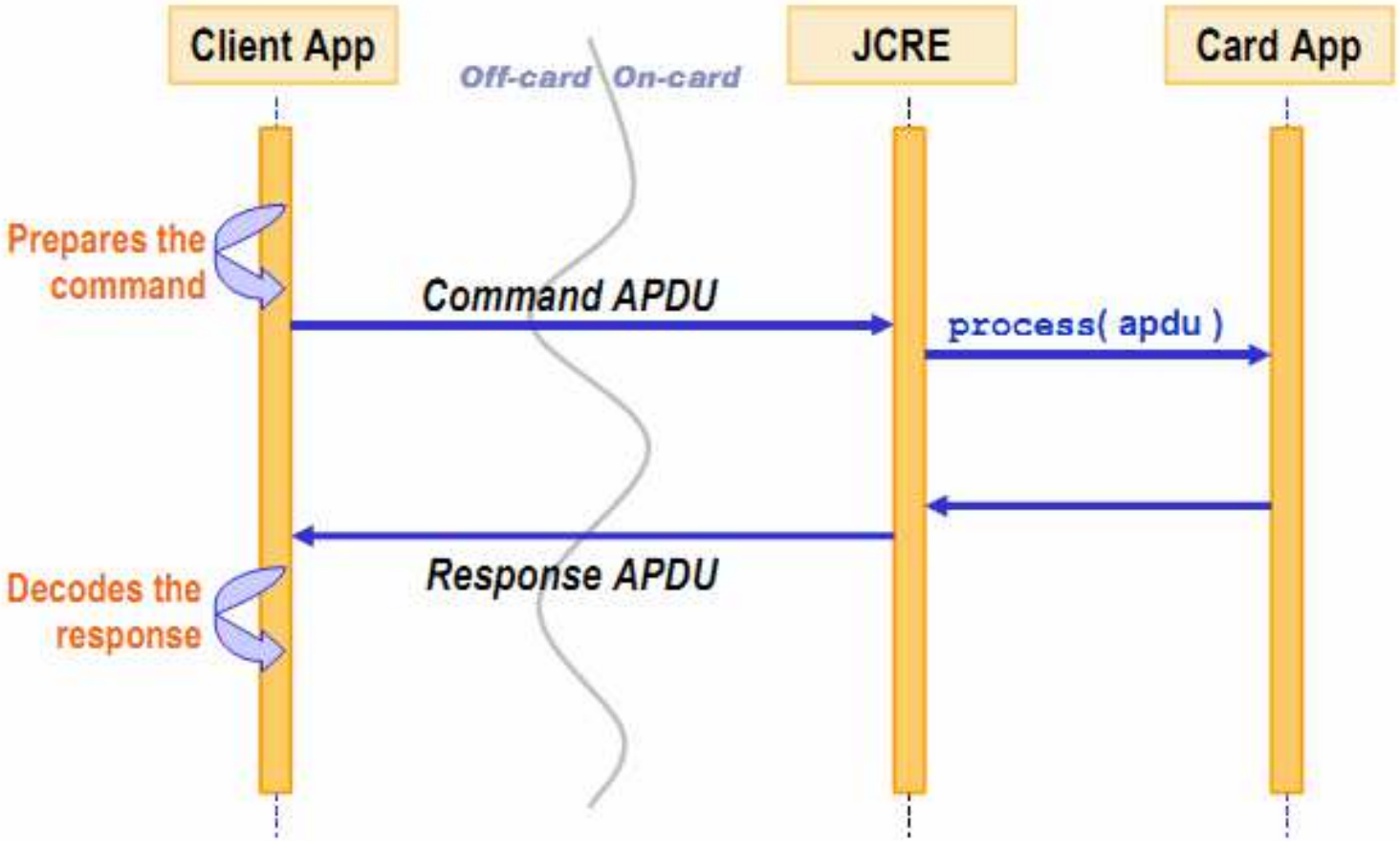
Role: Store a data that it receives and returns it to the terminal.

APDU command (xxxx)

APDU response (xxxx)

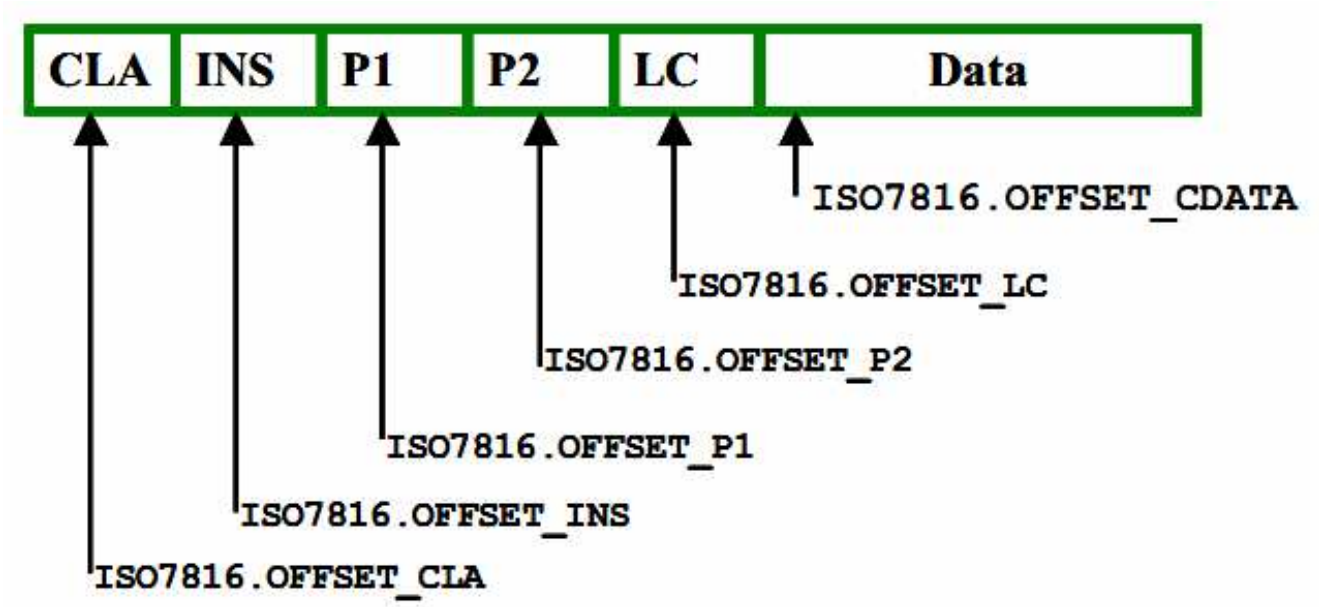
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card and the APDU Com./Resp



Source: Sebastian Hans, Java Card Platform overview, Sun Microsystems Inc., 2008

APDU command structure



Source: Sebastian Hans, Java Card Platform overview, Sun Microsystems Inc., 2008

le cnam

AIDs

- Define an AID for the package and an AID for the Applet

Package AID		
Field	**Value**	**Length**
RID	0xA0, 0x00, 0x00, 0x18, 0x50	5 octets
PIX	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x52, 0x41, 0x44, 0x50	10 octets (11 octets au max)
Applet AID		
Field	**Value**	**Length**
RID	0xF2, 0x34, 0x12, 0x34, 0x56	5 octets
PIX	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x52, 0x41, 0x44, 0x41	10 octets 11 octets au max)

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Applet methods

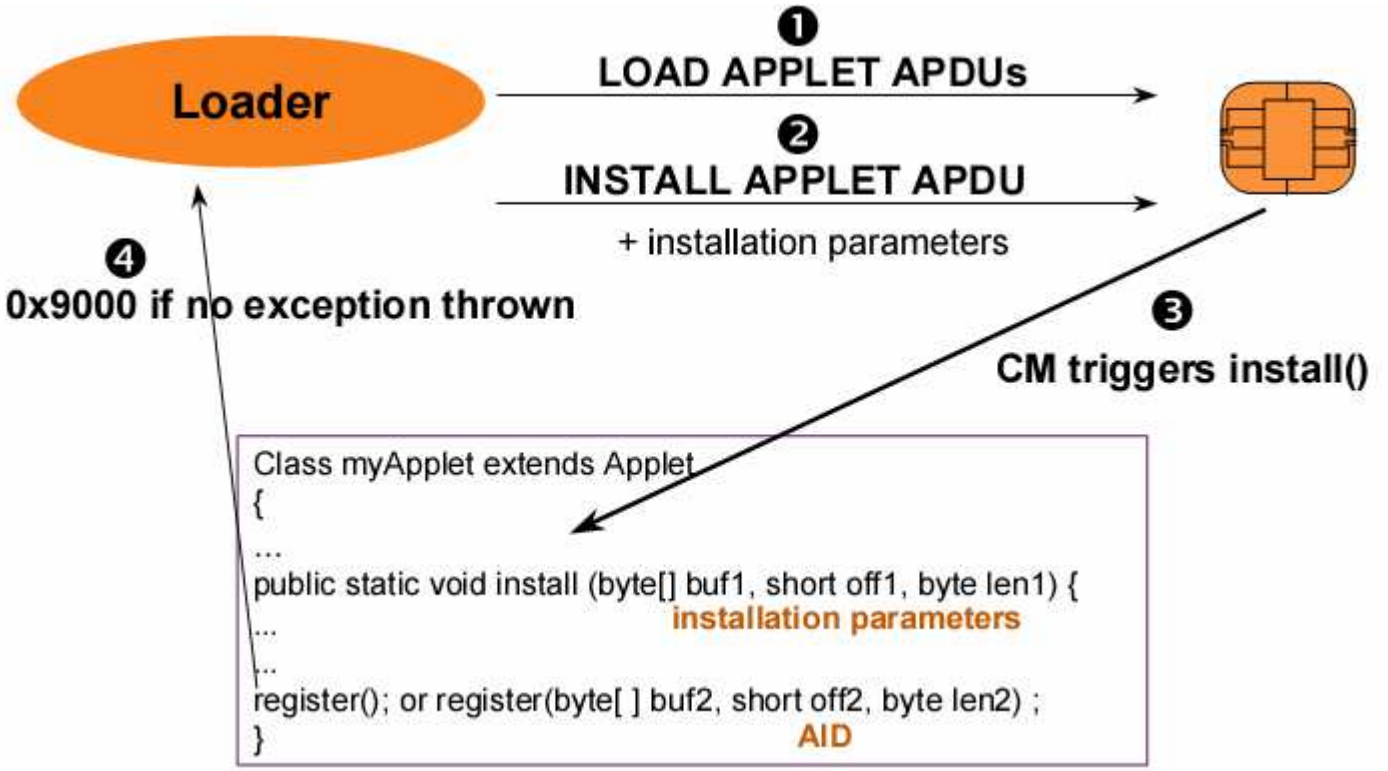
- An applet must always extend the class **`javacard.framework.Applet`**.

- The **`Applet`** class defines common methods to use to interact with the JCRE.

- These methods should be included in the body of the applet:
  - ✓Methods **`select/deselect`** : to activate/deactivate the applet
  - ✓Methods **`install/uninstall`**: to install/uninstall the applet sur la carte
  - ✓Method **`process`**: to process APDU commands and return APDU response
  - ✓Method **`register`**: to register the applet within the JCRE.

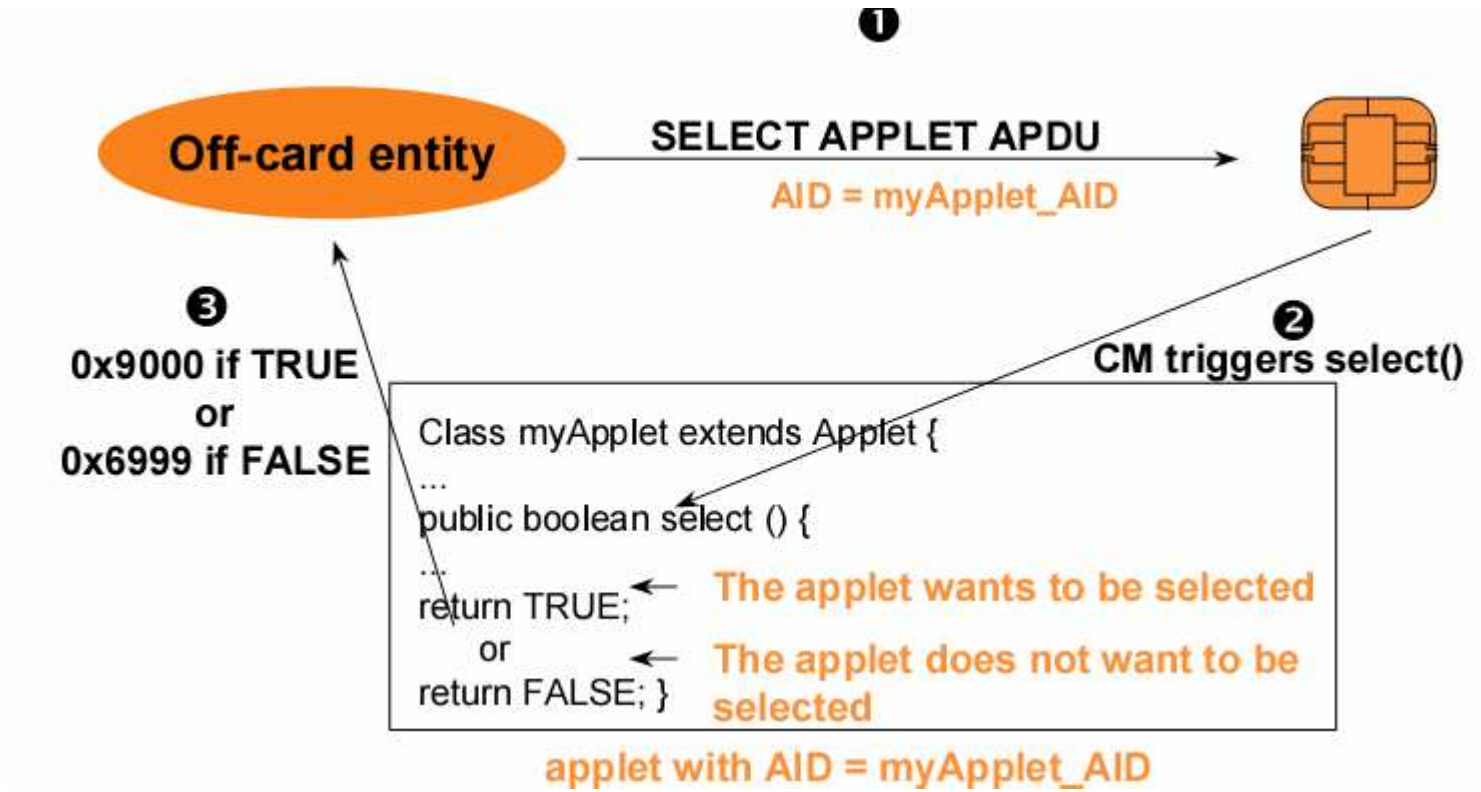
samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Install Method

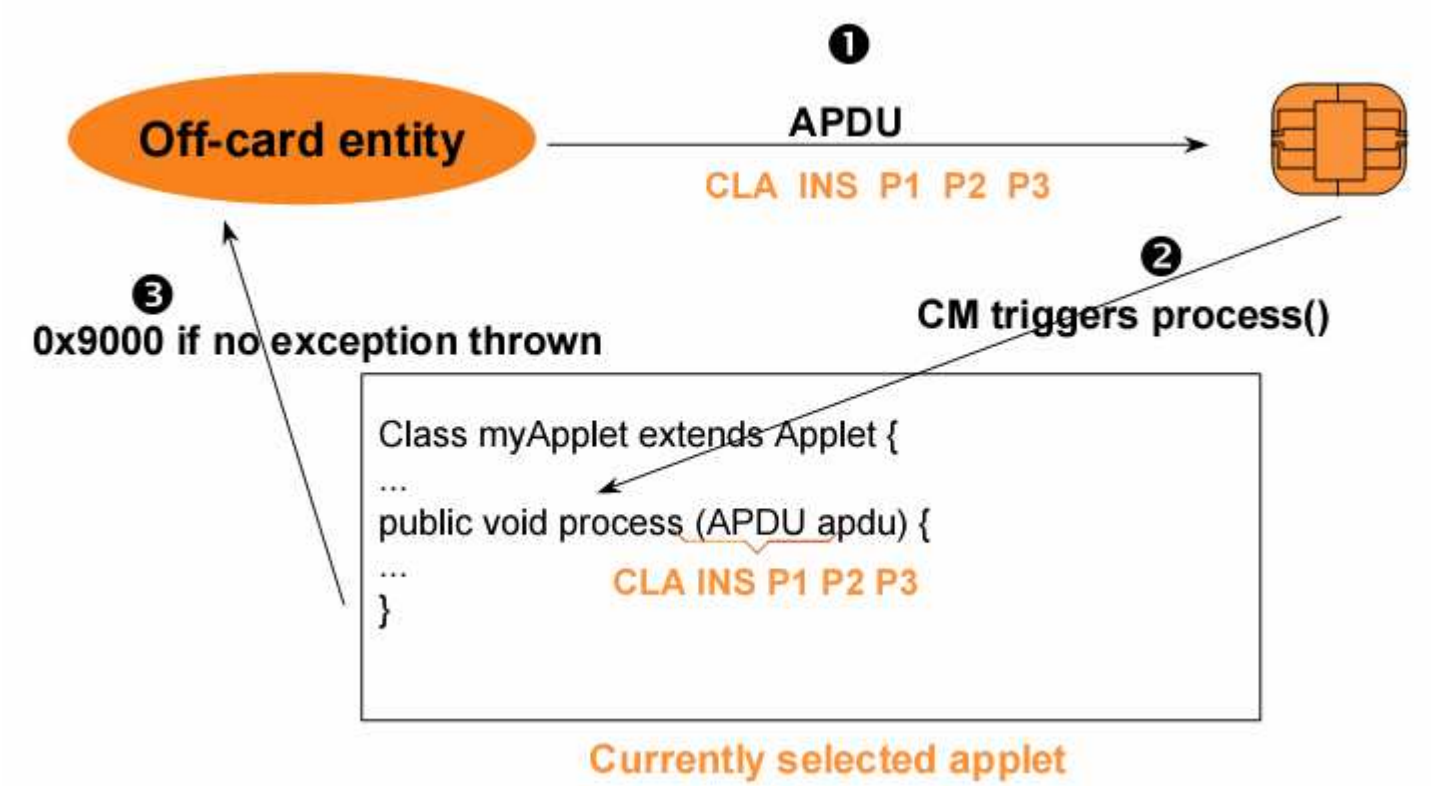


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

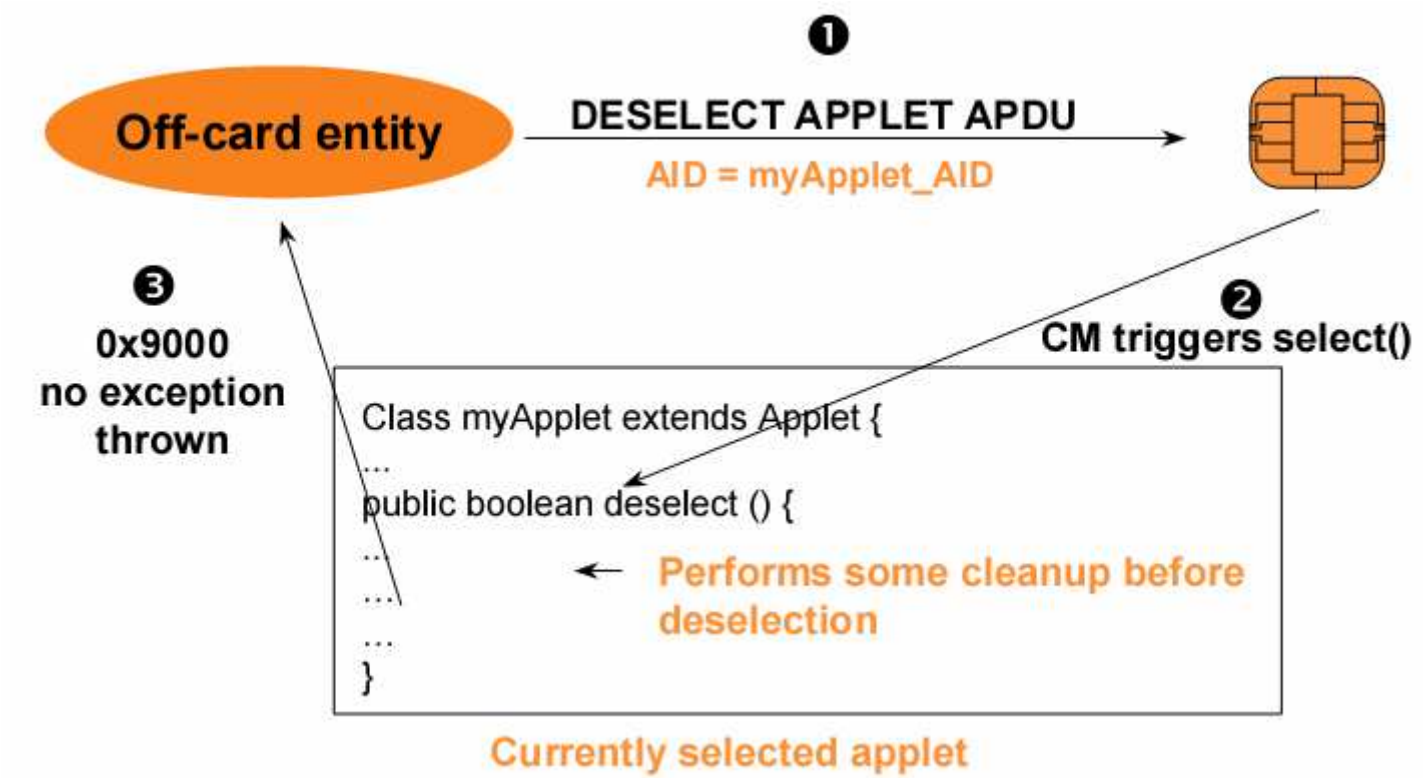
select Method



Process Method



deselect Method



samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

Methods to define in the applet

Method summary	
public void	deselect () Called by the JCRE to inform the currently selected applet that another (or the same) applet will be selected.
public Shareable	getShareableInterfaceObject (AID client AID, byte parameter) Called by the JCRE to obtain a sharable interface object from this server applet on behalf of a request from a client applet.
public static void	install (byte[] bArray, short bOffset, byte bLength) The JCRE calls this static method to create an instance of the Applet subclass.
public abstract void	process (APDU apdu) Called by the JCRE to process an incoming APDU command.
protected final void	register () This method is used by the applet to register this applet instance with the JCRE and assign the default AID in the CAD file to the applet instance.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

Methods to define in the applet

Method summary	
protected final void	register (byte[] bArray, short bOffset, byte bLength) This method is used by the applet to register this applet instance with the JCRE and to assign the specified AID in the array bArray to the applet instance.
public boolean	select () Called by the JCRE to inform this applet that it has been selected.
protected final boolean	selectingApplet () This method is used by the applet process() method to distinguish the SELECT APDU command that selected this applet from all other SELECT APDU APDU commands that may relate to file or internal applet state selection.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Purse Applet: example of code

```
public void PurseApplet (byte [] bArray, short bOffset, byte bLeng
       {
       balance = (short) 0x1000;
       register();
    }
    public static void install (byte [] bArray, short bOffset,
              byte bLength) {
       new PurseApplet (bArray, bOffset, bLength);
    }

public void process (APDU apdu) throws ISOException {
       byte [] buffer = apdu.getBuffer();
       …
```


samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

An interface between the applet and the terminal

➢**Define APDU commands :**

A Java Card applet must deal with a set of APDU commands :

✓SELECT APDU command: to select an applet on the card

✓ Processing APDUs: commands performed by the process () method

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Interface for the applet of electronic purse

Operations provided by the application:
Read the amount of the purse, debit or credit their account.

These operations are located on the applet using the following methods: **`getBalance (), credit (), debit ()`**
These methods are called directly by the **`process`** () method.

Hence: for each operation to define an APDU command triggers the corresponding method.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Commands of the electronic purse

APDU Command CREDIT						
APDU Command						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x30	0x0	0x0	1	The amount value to be credited	NS

APDU Command DEBIT						
APDU Command						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x40	0x0	0x0	1	The amount value to debit	NS

APDU Command GET BALANCE						
APDU command						
CLA	**INS**	**P1**	**P2**	**Lc**	**Data field**	**Le**
0xB0	0x50	0x0	0x0	NS	NS	2

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Processing APDU commands

1. Extract the APDU buffer

The applet invokes the method **`getBuffer`**() to extract the first 5 bytes available in the buffer: **`CLA, INS, P1, P2, et P3`**.

2. Receive data

If additional data in the command, the applet must invoke the method **`setIncomingAndReceive()`** to lead the APDU object to receive incoming data.

`receiveBytes`() allows to read the data.

3. Return data

`setOutgoing`() to get the length of the response (Le)

`setOutgoingLength`() to inform the CAD of the actual length of data to be returned.

`sendByteLong`() to send data from the buffer.

4. Return the word status.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Java Card 2.2.2 specifications

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Other functionalities of Java Card 2.2

- Logical channels
- Applet and package deletion
- Deletion of objects thanks to garbage collection
- Java Card Remote Method Invocation
- Support for AES and elliptic curves
- Support for contactless

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

API of Java Card 2.2.2

- **Package java.lang**
  - Arithmetic oOperations
  - Operations on arrays
  - Exception management,
  - etc.

- **Remote Method Invocation**

  - ✓**Package java.rmi**
    - Remote
    - RemoteException

  - ✓**Package javacard.framework.service**
    - BasicService
    - CardRemoteObject
    - Dispatcher
    - RemoteService
    - RMIService

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

API of Java Card 2.2.2

- **Package javacard.framework**
  - AID
  - APDU
  - APDUException
  - Applet
  - ISO7816
  - ISOException
  - JCSystem
  - MultiSelectable
  - OwnerPIN
  - PIN
  - PINException
  - Util
  - etc.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

API of Java Card 2.2.2

- **Package javacard.security**
  - AESKey
  - DESKey
  - DSAKey
  - DSAPrivateKey
  - DSAPublicKey
  - ECKey
  - ECPrivateKey
  - ECPublicKey
  - HMACKey
  - KeyBuilder
  - KoreanSEEDKey
  - RSAPrivateCrtKey
  - RSAPrivateKey
  - RSAPublicKey
  - etc.

- **Package javacardx.crypto**
  - Cipher
  - KeyEncryption

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Security Algorithms proposed in Java Card 2.2.2

- AES: Advanced Encryption Standard (FIPS-197)

- SEED Algorithm Specification : KISA - Korea Information Security Agency

Standard Names for Security and Crypto Packages
- SHA (SHA-1): Secure Hash Algorithm, as defined in Secure Hash Standard, NIST FIPS 180-1
- SHA-256,SHA-384,SHA-512: Secure Hash Algorithm,as defined in Secure Hash Standard,NIST FIPS 180-2
- MD5: The Message Digest algorithm RSA-MD5, as defined by RSA DSI in RFC 1321
- RIPEMD-160: as defined in ISO/IEC 10118-3:1998 Information technology – Security techniques - Hash-functions - Part 3: Dedicated hash-functions
- DSA: Digital Signature Algorithm, as defined in Digital Signature Standard, NIST FIPS 186
- DES: The Data Encryption Standard, as defined by NIST in FIPS 46-1 and 46-2
- RSA: The Rivest, Shamir and Adleman Asymmetric Cipher algorithm
- ECDSA: Elliptic Curve Digital Signature Algorithm
- ECDH: Elliptic Curve Diffie-Hellman algorithm
- AES: Advanced Encryption Standard (AES), as defined by NIST in FIPS 197
- HMAC: Keyed-Hashing for Message Authentication, as defined in RFC-2104

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

API of Java Card 2.2.2

- **Package javacardx.biometry**
  - BioBuilder
  - BioException
  - BioTemplate
  - OwnerBioTemplate
  - SharedBioTemplate

- **Package javacardx.framework.math**
  - BCDUtil
  - BigNumber
  - ParityBit

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

le cnam

API of Java Card 2.2.2

Core Packages	
java.io	Defines a subset of the java.io package in the standard Java programming language.
java.lang	Provides classes that are fundamental to the design of the Java Card technology subset of the Java programming language.
java.rmi	Defines the Remote interface which identifies interfaces whose methods can be invoked from card acceptance device (CAD) client applications.
javacard.framework	Provides a framework of classes and interfaces for building, communicating with and working with Java Card technology-based applets.
javacard.framework.service	Provides a service framework of classes and interfaces that allow a Java Card technology-based applet to be designed as an aggregation of service components.
javacard.security	Provides classes and interfaces that contain publicly-available functionality for implementing a security and cryptography framework on the Java Card platform.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

Standard Extensions	
javacardx.apdu	Extension package that enables support for ISO7816 specification defined optional APDU related mechanisms.
javacardx.biometry	Extension package that contains functionality for implementing a biometric framework on the Java Card platform.
javacardx.crypto	Extension package that contains functionality, which may be subject to export controls, for implementing a security and cryptography framework on the Java Card platform.
javacardx.external	Extension package that provides mechanisms to access memory subsystems which are not directly addressable by the Java Card runtime environment(Java Card RE) on the Java Card platform.
javacardx.framework.math	Extension package that contains common utility functions for BCD math and parity computations.
javacardx.framework.tlv	Extension package that contains functionality, for managing storage for BER TLV formatted data, based on the ASN.1 BER encoding rules of ISO/IEC 8825-1:2002, as well as parsing and editing BER TLV formatted data in I/O buffers.
javacardx.framework.util	Extension package that contains common utility functions for manipulating arrays of primitive components - byte, short or int.
javacardx.framework.util.intx	Extension package that contains common utility functions for using int components.

Conclusion

- Java Card offers the development of applications on cards with Java:
  - High level
  - With good properties (object-oriented concept).

- The loading infrastructure is defined by Global Platform

- The specifications of Java Card 3.0 has been published in March 2008 by Oracle
  - Classic Edition (extension of version 2.2)
  - Connected Edition (Web oriented)
  - Integration of TCP / IP stack, servlets, multi-threading, etc.

samia.bouzefrane@cnam.fr - CEDRIC (CNAM) -

References

1. http://java.sun.com/products/javacard/
2. http://java.sun.com/javacard/3.0/specs.jsp
2. Technology for smart cards: architecture and programmer's guide, Zhiqun Chen, Addison Wesley, sept. 2000
3. Understanding Java Card 2.0, Zhiqun Chen & Rinaldo Di Giorgio
4. http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html
5. http://javacardforum.org
6. Zhiqun Chen, "**How to write a Java Card applet: A developer's guide**", http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javacard.html.
7. Pierre Paradinas, Support de cours sur « **Java Card** », UV de Systèmes Enfouis et Embarqués, Valeur C, Laboratoire CEDRIC, CNAM. http://deptinfo.cnam.fr/~paradinas/cours/ValC-IntroJavaCard.pdf
8. **Global Platform, Card Specification :** http://www.globalplatform.org/specificationform2.asp?id=archived
9. **API Java Card** : http://java.sun.com/products/javacard/htmldoc
10. Eric Vétillard : http://javacard.vetilles.com/2006/09/17/hello-world-smart-card/
11. Training by Nemec from Gemalto, SIMAGINE, nov. 2007.