International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET

Java Program Running Smart Card

Swapnil A. Jamgade
Shree Radhika Tai Pandav College of Engineering and Technology, Nagpur University, Nagpur, Maharashtra, India

ARTICLEINFO

Article History:
Accepted: 01 June 2023
Published: 17 June 2023

Publication Issue
Volume 10, Issue 3
May-June-2023
Page Number
477-480

ABSTRACT

I have made this report file on the topic Java Card, I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic. My efforts and wholehearted co-corporation of each and every one has ended on a successful note. I express my sincere gratitude who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Keywords : Java Card, Development Kit, APDU Tool

I. INTRODUCTION

Java Card is a smart card that is capable of running programs written in Java. For this a new Java platform, Sun's JavaSoft division has made available the Java Card 2.0 API specification, and several licensees are now implementing this API on smart cards. In order to program Java Cards that are 2.0-compliant, developers need to understand what a Java Card is architecturally, what its core classes are, and how to develop applications for the card. This article gets inside a Java Card, providing you, the developer, with technical guidance on the system architecture, application programming interface, and runtime environment of the Java platform in a smart card.

II. METHODS AND MATERIAL

2.a.1) WHAT IS A JAVA CARD?

A Java Card is a smart card that can run Java programs. The Java Card 2.0 specification contains detailed information for building the Java Card virtual machine and application programming interface (API) in smart cards. The minimum system requirement is 16 kilobytes of read-only memory (ROM), 8 kilobytes of EEPROM, and 256 bytes of random-access memory (RAM).

2.a.2) The lifetime of a Java Card

The Java Card lifetime starts when the native OS, Java Card VM, API classes libraries and optionally, applets are burned into ROM. This process of writing the permanent components into the non-mutable memory of a chip for carrying out incoming commands is called masking. Before it lands in your wallet, a Java Card needs to go through initialization and personalization. Initialization refers to loading general data into a card's non-volatile memory. This data is identical across a large number of cards and is not specific to an individual; an example might be the issuer or manufacture's name.

2.a.3) Lifetime of a Java Card virtual machine
Unlike the Java virtual machine (JVM) in a PC or workstation, the Java Card virtual machine runs forever. Most of the information stored on the card must be preserved even when the power is removed -- that is, when the card is removed from the reader. The Java Card VM creates objects in EEPROM to hold the persistent information. The execution lifetime of the Java Card VM is the lifetime of the card. When the power is not provided, the VM runs in an infinite clock cycle.

2.a.4) The lifetime of Java Card applets and objects
An applet's life starts when it is properly installed and registered with the system's registry table and ends when it is removed from the table. The space of a removed applet may or may not be reused, however, depending on whether garbage collection is implemented on the card. An applet on a card is in an inactive stage until it is explicitly selected by the terminal.

2.B.1) Java Card 2.0 language subset
Java Card programs are, of course, written in Java. They are compiled using common Java compilers. Due to limited memory resources and computing power, not all the language features defined in the Java Language Specification are supported on the Java Card. Specifically, the

Java Card does not support:
• Dynamic class loading
• Security manager
• Threads and synchronization
• Object cloning
• Finalization
• Large primitive data types (float, double, long, and char) It's no surprise that keywords that support those features are also omitted from the language. VM implementers may decide to support 32-bit integer type or native methods for post-issuance applets if they are working on a more advanced smart card with more memory. Post-issuance applets are those applets that are installed on a Java Card after the card is issued to a card holder

2.B.2The Java Card 2.0 framework
Smart cards have been in the market for 20 years, and most of them are generally compatible with ISO 7816 Parts 1-7 and/or EMV. We've already looked at ISO 7816. What's EMV? The EMV standard, defined by Europay, MasterCard, and Visa, is based on the ISO 7816 series of standards with additional proprietary features to meet the specific needs of the financial industry. The Java Card Framework is designed to easily support smart card systems and applications. It hides the details of the smart card infrastructure and provides Java Card application developers with a relatively easy and straightforward programming interface

III. DEVELOPING A JAVA CARD APPLET

After you write a Java Card applet, you're ready to prepare it for execution in a Smart Card that implements the Java Card runtime environment. Preparing a Java Card applet for execution involves a number of steps, such as converting it to a runtime format and testing it in various simulated environments. Using the Java Card

Development Kit Use the Java Card 2.1.2
Development Kit to develop a Java Card applet. You can use the Java Card 2.1.2 Development Kit to develop an applet for masking. Masking means embedding the applet into the read-only memory of a smart card when the card is manufactured. Alternatively, you can use the Java Card 2.1.2 Development Kit to develop an applet for installation onto a smart card after the card is manufactured. The Java Card 2.1.2

Development Kit provides components and tools that you need to develop applets for masking or installation.

This includes:
• Java Card Framework classes that are essential for developing Java Card applets.
• A Java Card Workstation Development Environment (JCWDE) that simulates the Java Card runtime environment on a Java[tm] virtual machine.
• An APDUTool utility that sends command APDUs to the JCWDE or to a Java Card runtime environment. Command APDUs are the way operational requests are made to a smart card.
• A Converter tool that converts a Java Card applet into a format required for masking or for installation. • Off-card verification tools that check the integrity of files produced by the Converter.
• A mask generator that generates a mask file for incorporation into a mask in a Java Card runtime environment.
• An off-card installer for installing a Java Card applet onto a smart card.
• Using these classes and tools, you develop a Java Card applet on your workstation or PC. Specifically, you:
• Compile the applet.
• Optionally, test the applet in the JCWDE, and debug the applet.
• Convert the applet. If you develop an applet that will be masked, you convert the applet class and all the classes in its package to a Java Card Assembly (JCA) file. If you develop an applet for installation, you convert the applet and all the classes in its package to a Converted Applet (CAP) file, and possibly an export file. An export file is used to convert another package if that package imports classes from this package. The next step depends on whether you develop an applet for masking or for installation. For masking, you run the mask generator to produce a mask file. For installation, you run the off-card installer; this produces a script file that contains command APDUs -- you then use the file as input to the APDU Tool. The APDU Tool works in conjunction with the installer on the smart card to download the CAP file and instantiate the Java Card applet in the CAP file.

IV. RESULTS AND DISCUSSION

4.a.1) Compiling a Java Card Applet
You write Java Card applets in the Java programming language. However because applets are designed to run in the very small memory space of a smart card, they're coded using an appropriate subset of the Java programming language. As you do for a Java application or applet, you compile Java Card applets on your workstation or PC. You can use any Java compiler that supports Java 2 Platform, Standard Edition version 1.2.2, 1.3 (or above), such as the javac compiler in Java 2 SDK version 1.3. Remember to include api21.jar in your class path before you compile ava Card will require marketing promotion, e applications and tools development, and At the same time, the number of Java in existence could easily extend into the ions within the next few years. Which means may soon be storing your personal rmation and downloading applications using card you carry around in your wallet or purse.

4.a.2Debugging a Java Card Applet

You can debug the applet on your workstation or PC just as you do for a Java application. More specifically, you can use the same debugging tools, such as the debugging facilities of an IDE, or the Java debugger tool (jdb) in the Java 2 SDK. Converting a Java Card Applet In Java Card technology, you don't directly incorporate a Java Card applet into a mask. Similarly, after a smart card is manufactured, you don't directly download a Java Card applet for installation onto a smart card. Instead, for masking, you convert an applet class and all the classes in its package to a JCA (Java Card Assembly) file. The JCA file and JCA files for any other packages to be included in the mask are then converted into a format compatible with the target runtime environment. It's this converted output for the target runtime environment that is incorporated into the mask

V. CONCLUSION

Java Card can be used in all fields where the smart card is now being used. Java Card can be used as an ID card which contains personal information, as a medical card which stores medical information, credit/debit bank card, as an electronic purse etc. Multi-Application Java Cards, that is, more than one application in a single card is also available. The Java Card adds a new platform to the world of Java. Widespread adoption and deployment

VI. REFERENCES

[1]. Pascal Urien, "Cloud of secure elements: An infrastructure for the trust of mobile NFC services", 2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp.213-218, 2014.

[2]. Pascal Urien, Selwyn Piramuthu, "Towards a secure Cloud of Secure Elements concepts and experiments with NFC mobiles", 2013 International Conference on Collaboration Technologies and Systems (CTS), pp.166-173, 2013.

[3]. Dawei Zhang, "Design and Implementation of SMS4 on Java Card", 2009 WRI World Congress on Software Engineering, vol.1, pp.145-149, 2009.

[4]. Ula M. Qabs, Fawzi M. Al-Naima, "Design and implementation of a Smart Card Simulator", 2008 International Conference on Computer and Communication Engineering, pp.217-220, 2008.

[5]. Yoon-sim Yang, Won-ho Choi, Min-sik Jin, Cheul-jun Hwang, Min-soo Jung, "An Advanced Java Card System Architecture for Smart Card Based on Large RAM Memory", 2006 International Conference on Hybrid Information Technology, vol.2, pp.646-650, 2006.

[6]. R. Thibadeau, "Trusted Computing for Disk Drives and Other Peripherals", IEEE Security & Privacy, vol.4, no.5, pp.26-33, 2006.

Cite this article as :

Swapnil A. Jamgade, "Java Program Running Smart Card", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 10 Issue 3, pp. 477-480, May-June 2023.
Journal URL : https://ijsrset.com/IJSRSET23103141