Java Card

Java Card is a software technology that allows Java-based applications (applets) to be run securely on smart cards and more generally on similar secure small memory footprint devices[1] which are called "secure elements" (SE). Today, a secure element is not limited to its smart cards and other removable cryptographic tokens form factors; embedded SEs soldered onto a device board and new security designs embedded into general purpose chips are also widely used. Java Card addresses this hardware fragmentation and specificities while retaining code portability brought forward by Java.

Java Card

Player software	Java
Programming language(s)	Java
Application(s)	Smart card
Status	Active
License	Proprietary licence by Oracle
Website	www.oracle.com/java/java-card/ (https://www.oracle.com/java/java-card/)

Java Card is the tiniest of Java platforms targeted for embedded devices. Java Card gives the user the ability to program the devices and make them application specific. It is widely used in different markets: wireless telecommunications within SIM cards and embedded SIMs (eSIM), payment within banking cards[2] and RFID (NFC) mobile payment and for identity cards, healthcare cards, and passports. Several IoT products like gateways are also using Java Card based products to secure communications with a cloud service for instance.

The first Java Card was introduced in 1996 by Schlumberger's card division which later merged with Gemplus to form Gemalto. Java Card products are based on the specifications by Sun Microsystems (Defunct in 2010 as it was acquired by Oracle Corporation). Many Java card products also rely on the GlobalPlatform specifications for the secure management of applications on the card (download, installation, personalization, deletion).

The main design goals of the Java Card technology are portability, security and backward compatibility.[3]

Portability

Java Card aims at defining a standard smart card computing environment allowing the same Java Card applet to run on different smart cards, much like a Java applet runs on different computers. As in Java, this is accomplished using the combination of a virtual machine (the Java Card Virtual Machine), and a well-defined runtime library, which largely abstracts the applet from differences between smart cards. Portability remains mitigated by issues of memory size, performance, and runtime support (e.g. for communication protocols or cryptographic algorithms). Moreover, vendors often expose proprietary APIs specific to their ecosystem, further limiting portability for applets that rely on such calls. To address these limitations, Vasilios Mavroudis and Petr Svenda introduced JCMathLib, an open-source cryptographic wrapper library for Java Card, enabling low-level cryptographic computations not supported by the standard API.[4][5][6]

Security

Java Card technology was originally developed for the purpose of securing sensitive information stored on smart cards. Security is determined by various aspects of this technology:

Data encapsulation
: Data is stored within the application, and Java Card applications are executed in an isolated environment (the Java Card VM), separate from the underlying operating system and hardware.

Applet firewall
: Unlike other Java VMs, a Java Card VM usually manages several applications, each one controlling sensitive data. Different applications are therefore separated from each other by an applet firewall which restricts and checks access of data elements of one applet to another.

Cryptography
: Commonly used symmetric key algorithms like DES, Triple DES, AES, and asymmetric key algorithms such as RSA, elliptic curve cryptography are supported as well as other cryptographic services like signing, key generation and key exchange.

Applet
: The applet is a state machine which processes only incoming command requests and responds by sending data or response status words back to the interface device.

Design

At the language level, Java Card is a precise subset of Java: all language constructs of Java Card exist in Java and behave identically. This goes to the point that as part of a standard build cycle, a Java Card program is compiled into a Java class file by a Java compiler; the class file is post-processed by tools specific to the Java Card platform.

However, many Java language features are not supported by Java Card (in particular types `char`, `double`, `float` and `long`; the `transient` qualifier; enums; arrays of more than one dimension; finalization; object cloning; threads). Further, some common features of Java are not provided at runtime by many actual smart cards (in particular type `int`, which is the default type of a Java expression; and garbage collection of objects).

Bytecode

Java Card bytecode run by the Java Card Virtual Machine is a functional subset of Java 2 bytecode run by a standard Java Virtual Machine but with a different encoding to optimize for size. A Java Card applet thus typically uses less bytecode than the hypothetical Java applet obtained by compiling the same Java source code. This conserves memory, a necessity in resource constrained devices like smart cards. As a design tradeoff, there is no support for some Java language features (as mentioned above), and size limitations. Techniques exist for overcoming the size limitations, such as dividing the application's code into packages below the 64 KiB limit.

Library and runtime

Standard Java Card class library and runtime support differs a lot from that in Java, and the common subset is minimal. Most packages offered in the Java SE standard library are not present. For example, the Java Security Manager class is not supported in Java Card, where security policies are implemented by the Java

Card Virtual Machine; and transients (non-persistent, fast RAM variables that can be class members) are supported via a Java Card class library, while they have native language support in Java. For example, Java Card has no `String` type, only `byte[]` (similar to C, where C strings are represented as `char[]`). Most primitives are absent except for `byte`, `short`, `int` and `boolean`.[7]

`java.*`

The `java.lang` package contains fundamental classes of the Java Card technology, which include the two classes `Object` (the root of the class hierarchy), `Throwable` (which has only one known subclass `Exception`, as `Error` is not present) and relevant exceptions. As mentioned earlier, `java.lang.String` does not exist in Java Card due to the limitations of the platform, instead representing strings as `byte[]`. Arrays (`T[]`) have the `length` field.

There are no wrapper classes `Byte`, `Short`, `Integer` and `Boolean` for the primitive types `byte`, `short`, `int` and `boolean`.

All exceptions except `java.io.IOException` and `javacard.framework.CardException` are instances of `java.lang.RuntimeException`.

The package `java.rmi` features the interface `Remote` for methods which are invoked from a card acceptance device (CAD) client application.

`javacard.*`

The `javacard.framework` package contains the framework of classes for creating Java Card applets. The `Applet` is an abstract class for an applet. The `Util` class contains utility methods on byte arrays (`byte[]`) and `short`s.

The `javacard.security` package contains interfaces and classes for working with cryptographic and security functionality.

`javacardx.*`

These are extension packages to supplement the `javacard.*` packages.

The `javacardx.framework.*` packages contain various utilities, such as class `javacardx.framework.math.BigNumber` (operations on Big Numbers), `javacardx.framework.nio` (a package for buffers, containers for data), `javacardx.framework.string.StringUtil` (a class providing methods on UTF-8 encoded strings, still represented as `byte[]`), package `javacardx.framework.time` which contains classes `SysTime` (for system time) and `TimeDuration` (for representing an amount of time), and `javacardx.framework.util.intx.JCint` for working with `int`.

The `javacardx.security.*` packages contains functionality for working with certificates, cryptographic derivation functions, and more.

The `javacardx.biometry` contains functionality for biometric frameworks on Java Card. The class `javacardx.biometry.BioBuilder` is used for creating biometric reference templates, and contains various constants representing different biometric identification means.

Specific features

The Java Card runtime and virtual machine also support features that are specific to the Java Card platform:

Persistence
: With Java Card, objects are by default stored in persistent memory (RAM is very scarce on smart cards, and it is only used for temporary or security-sensitive objects). The runtime environment as well as the bytecode have therefore been adapted to manage persistent objects.

Atomicity
: As smart cards are externally powered and rely on persistent memory, persistent updates must be atomic. The individual write operations performed by individual bytecode instructions and API methods are therefore guaranteed atomic, and the Java Card Runtime includes a limited transaction mechanism.

Applet isolation
: The Java Card firewall is a mechanism that isolates the different applets present on a card from each other. It also includes a sharing mechanism that allows an applet to explicitly make an object available to other applets.

Development

Coding techniques used in a practical Java Card program differ significantly from those used in a Java program. Still, that Java Card uses a precise subset of the Java language speeds up the learning curve, and enables using a Java environment to develop and debug a Java Card program (caveat: even if debugging occurs with Java bytecode, make sure that the class file fits the limitation of Java Card language by converting it to Java Card bytecode; and test in a real Java Card smart card early on to get an idea of the performance); further, one can run and debug both the Java Card code for the application to be embedded in a smart card, and a Java application that will be in the host using the smart card, all working jointly in the same environment.

Example

This is a "Hello, World!" program example in Java Card.

```
package com.wikipedia.example;

import javacard.framework.APDU;
import javacard.framework.Applet;
import javacard.framework.ISO7816;
import javacard.framework.ISOException;
import javacard.framework.Util;

public class HelloWorldApplet extends Applet {
    // INS (instruction) byte for "say hello"
    final static byte HELLO_INS = (byte)0x01;

    // AID (Application Identifier) for the applet
    private static final byte[] HELLO_APPLET_AID = {
        (byte)0xA0, (byte)0x00, (byte)0x00, (byte)0x00,
        (byte)0x62, (byte)0x03, (byte)0x01, (byte)0x0C, (byte)0x01
    };

    protected HelloWorldApplet() {
        register();
    }

    public static void install(byte[] bArray, short bOffset, byte bLength) {
        new HelloWorldApplet();
    }

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

```
        if (selectingApplet()) {
            return;
        }

        byte ins = buffer[ISO7816.OFFSET_INS];

        if (ins == HELLO_INS) {
            sendHello(apdu);
        } else {
            ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
    }

    private void sendHello(APDU apdu) {
        byte[] buffer = apdu.getBuffer();
        byte[] hello = { 'H', 'e', 'l', 'l', 'o' };

        short length = (short)hello.length;
        Util.arrayCopyNonAtomic(hello, (short)0, buffer, (short)0, length);
        apdu.setOutgoingAndSend((short)0, length);
    }
}
```

Versions

Oracle has released several Java Card platform specifications and is providing SDK tools for application development. Usually smart card vendors implement just a subset of algorithms specified in Java Card platform target and the only way to discover what subset of specification is implemented is to test the card.[8]

- Version 3.2 (30.01.2023)[9]
  - Introduced support for (D)TLS1.3 protocols
  - Added API clarifications to help application developers and significantly increase the level of interoperability across multiple implementations
- Version 3.1 (17.12.2018)[10]
  - Added configurable key pair generation support, named elliptic curves support, new algorithms and operations support, additional AES modes and Chinese algorithms.
- Version 3.0.5 (03.06.2015)
  - Oracle SDK: Java Card Classic Development Kit 3.0.5u1 (03.06.2015)
  - Added support for Diffie-Hellman modular exponentiation, Domain Data Conservation for Diffie-Hellman, Elliptic Curve and DSA keys, RSA-3072, SHA3, plain ECDSA, AES CMAC, AES CTR.
- Version 3.0.4 (06.08.2011)
  - Oracle SDK: Java Card Classic Development Kit 3.0.4 (06.11.2011)
  - Added support for DES MAC8 ISO9797.
- Version 3.0.1 (15.06.2009)
  - Oracle SDK: Java Card Development Kit 3.0.3 RR (11.11.2010)
  - Added support for SHA-224, SHA-2 for all signature algorithms.
- Version 2.2.2 (03.2006)
  - Oracle SDK: Java Card Development Kit 2.2.2 (03.2006)
  - Added support for SHA-256, SHA-384, SHA-512, ISO9796-2, HMAC, Korean SEED MAC NOPAD, Korean SEED NOPAD.

- Version 2.2.1 (10.2003)
  - Oracle SDK: Java Card Development Kit 2.2.1 (10.2003)
- Version 2.2 (11.2002)
  - Added support for AES cryptography key encapsulation, CRC algorithms, Elliptic Curve Cryptography key encapsulation,Diffie-Hellman key exchange using ECC, ECC keys for binary polynomial curves and for prime integer curves, AES, ECC and RSA with variable key lengths.
- Version 2.1.1 (18.05.2000)
  - Oracle SDK: Java Card Development Kit 2.1.2 (05.04.2001)
  - Added support for RSA without padding.
- Version 2.1 (07.06.1999)

Java Card 3.0

The version 3.0 of the Java Card specification (draft released in March 2008) is separated in two editions: the *Classic Edition* and the *Connected Edition*.[11]

- The *Classic Edition* (currently at version 3.0.5 released in June 2015) is an evolution of the Java Card Platform version 2 (which last version 2.2.2 was released in March 2006), which supports traditional card applets on resource-constrained devices such as Smart Cards. Older applets are generally compatible with newer Classic Edition devices, and applets for these newer devices can be compatible with older devices if not referring to new library functions. Smart Cards implementing Java Card Classic Edition have been security-certified by multiple vendors, and are commercially available.
- The *Connected Edition* (currently at version 3.0.2 released in December 2009) aims to provide a new virtual machine and an enhanced execution environment with network-oriented features. Applications can be developed as classic card applets requested by APDU commands or as servlets using HTTP to support web-based schemes of communication (HTML, REST, SOAP ...) with the card. The runtime uses a subset of the Java (1.)6 bytecode, without Floating Point; it supports volatile objects (garbage collection), multithreading, inter-application communications facilities, persistence, transactions, card management facilities ... As of 2021, there has been little adoption in commercially available Smart Cards, so much that reference to Java Card (including in the present Wikipedia page) often implicitly excludes the *Connected Edition*.

Java Card 3.1

Java Card 3.1 was released in January 2019.

New CAP file Format and Applet Deployment Model

- Applet functionality can be split into multiple Java packages
- CAP file sizes can exceed 64KB

New I/O Framework and Trusted Peripherals

- A variety of physical layers and application protocol is supported, beyond smart card protocols defined in ISO 7816
- Logical access to device peripherals by secure element applications is facilitated

Core Platform Enhancements

- Array Views (views on a subset of an array), Static Resources embedded within a CAP file and Improved API extensibility

Security Services

- Certificate API, Key Derivation API, Monotonic Counter API, System Time API

New Cryptographic Extensions

- Configurable Key Pair generation, Named Elliptic Curves like Edwards-Curves, Additional AES modes (CFB & XTS), Chinese Algorithms (SM2 - SM3 - SM4)

See also

Computer programming portal

- Java Card OpenPlatform
- Tiny C Compiler, a similar subset of a larger language

References

1. Chen, Z. (2000). *Java Card Technology for Smart Cards: Architecture and Programmer's Guide* (https://archive.org/details/javacardtmtechno00zhiq). Addison-Wesley Java Series. Addison-Wesley. ISBN 978-0-201-70329-0. Retrieved 9 April 2019.
2. Oracle Learning Library (2013-01-30), *Developing Java Card Applications* (https://www.youtube.com/watch?v=khgT5dwKvOo), archived (https://ghostarchive.org/varchive/youtube/20211213/khgT5dwKvOo) from the original on 2021-12-13, retrieved 2019-04-18
3. Ahmed Patel; Kenan Kalajdzic; Laleh Golafshan; Mona Taghavi (2011). "Design and Implementation of a Zero-Knowledge Authentication Framework for Java Card" (http://www.igi-global.com/article/international-journal-information-security-privacy/58979). *International Journal of Information Security and Privacy*. **5** (3). IGI: 1–18. doi:10.4018/ijisp.2011070101 (https://doi.org/10.4018%2Fijisp.2011070101).
4. Mavroudis, Vasilios; Svenda, Petr (2020). "JCMathLib: Wrapper Cryptographic Library for Transparent and Certifiable JavaCard Applets". *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. pp. 383–390. arXiv:2008.11362 (https://arxiv.org/abs/2008.11362). doi:10.1109/EuroSPW51379.2020.00056 (https://doi.org/10.1109%2FEuroSPW51379.2020.00056).
5. "JCMathLib" (https://github.com/OpenCryptoProject/JCMathLib). *GitHub*. Retrieved 2025-04-12.
6. "OpenCrypto: Unchaining the JavaCard Ecosystem" (https://www.youtube.com/watch?v=vd0-Uhx2OoQ). *YouTube*. Retrieved 2025-04-12.
7. "Overview (Java Card API, Classic Edition)" (https://docs.oracle.com/en/java/javacard/3.2/jcapi/api_classic/index.html). *docs.oracle.com*. Oracle Corporation. Retrieved 13 October 2025.
8. "JCAlgTest - database of supported JavaCard algorithms" (http://www.fi.muni.cz/~xsvenda/jcsupport.html). Retrieved 27 January 2016.
9. Ponsini, Nicolas (30 January 2023). "Announcing Java Card 3.2 Release" (https://blogs.oracle.com/java/post/announcing-java-card-32-release). *Java Card Blog*. Retrieved 6 February 2023.
10. Ponsini, Nicolas. "Unveiling Java Card 3.1: New Cryptographic Extensions" (https://blogs.oracle.com/javaiot/unveiling-java-card-31%3A-new-cryptograpic-extensions). *blogs.oracle.com*.

Retrieved 2019-04-18.

11. Samoylov, N. (2018). *Introduction to Programming: Learn to program in Java with data structures, algorithms, and logic* (https://books.google.com/books?id=lOpgDwAAQBAJ&pg=PA13). Packt Publishing. p. 13. ISBN 978-1-78883-416-2. Retrieved 9 April 2019.

External links

- Java Card overview (https://www.oracle.com/java/technologies/java-card-tech.html) (Oracle)
- Defcon 21: The Secret Life of SIM Cards (https://www.youtube.com/watch?v=31D94QOo2gY) on YouTube
- JavaCards-OpenSC (https://github.com/OpenSC/OpenSC/wiki/JavaCards)
-