Java Card Technology

Prepared by: Ali Toyserkani

Adopted from:
“Introduction to Java Card Technology”
C. Enrique Ortiz
Presentation Agenda

- Identify Java Card Technology
- Identify Elements of Java Card Applications
- Communicating with a Java Card Applet
- Java Card Language Limitations
- Questions
Java Card?

What is a Java Card?

Simple Answer:

- Java + Card = Java Card !!
- Adoption of Java Platform for usage on Smart Cards
Smart Cards?

- A smart card is a plastic card that contains an embedded integrated circuit (IC)

- Examples:
  - Our very Own T-Card!
  - Credit Cards
  - Cell Phone SIM Cards
  - ...
Smart Cards are good!? How?

- They store and process Information
- Smart Cards Can be used to add authentication and secure access to information systems that require a high level of security
Properties of Smart Cards:

- Highly secure - Tempering with one, results in destruction of the information it contains
- Don’t contain battery - Become active when connected with a card reader
- Come in two forms Contact or Contact less
Contact or Contact Less?

- contact smart cards work by communicating via physical contact between a card reader and the smart card
- contact less smart cards communicate by means of a radio frequency signal, with a typical range of less than 2 feet
Now that we know Smart Cards:

- How does a smart card operate?
- What’s the platform used for its operations?
- What are different elements of these smart cards?
Elements of Java Card Application:

- A complete Java Card application consists of:
  - a back-end application
  - a host (off-card) application
  - an interface device (card reader)
  - the on-card applet
  - user credentials
  - and supporting software
Elements of Java Card Application:

- Back-End Application and System(s)
- Reader-Side
  - Host Application
  - Card Acceptance Device
  - Commands and Responses
- Card-Side
  - Applet
  - Applet
  - Applet
  - Vendor- or Industry-Specific Extensions
  - Java Card Framework and APIs
  - Java Card VM
  - Card OS
  - Java Card Runtime Environment
The Back-End Application and Systems

- Provide connectivity to security systems

- Example:
  - In an electronic payment system, the back-end application could provide access to credit card and other payment information
The Reader-Side

- Consists of two parts:
  - Host Application
  - Card Acceptance Device

- Think of a bank machine:
  - Host Application as the Computer that provides interaction with the system
  - Card Acceptance Device being where you put your debit card in
The Card-Side

- **Elements:**
  - One or more Java Applets
  - Card’s operating System
  - Java Card Runtime Environment (JCRE)
    - Java Card Virtual Machine
    - Java Card Framework and APIs
Communicating with a Java Card Applet

- Two methods for communicating with Java Card Applet:
  1. Fundamental message-passing model
  2. Java Card Remote Method Invocation (JCRMI) which is a subset of J2SE RMI!
The Message-Passing Model

Reader-Side

Host Application

Card-Side

Command APDU

T=0, T=1, T=USB, or T=RF Link Protocol

Response APDU

Applet

Applet

Applet

Java Card Framework and APIs

Java Card VM
The Message-Passing Model

- All Java Card applets extend the Applet base class and must implement the `install()` and `process()` methods.
- JCRE calls `install()` when installing the applet, and `process()` every time there is an incoming APDU for the applet.
- APDU: a logical data packet that's exchanged between the CAD and the Java Card Framework (It is considered as the center piece for the Message-Passing Model).
The Message-Passing Model

1. The Command APDU

<table>
<thead>
<tr>
<th>Command APDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header (required)</td>
</tr>
<tr>
<td>CLA</td>
</tr>
<tr>
<td>INS</td>
</tr>
<tr>
<td>P1</td>
</tr>
<tr>
<td>P2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Body (optional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lc</td>
</tr>
<tr>
<td>Data Field</td>
</tr>
<tr>
<td>Le</td>
</tr>
</tbody>
</table>
The Message-Passing Model

2. The Response APDU

The format of a response APDU is much simpler:

<table>
<thead>
<tr>
<th>Response APDU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Body (optional)</strong></td>
</tr>
<tr>
<td><strong>Data Field</strong></td>
</tr>
</tbody>
</table>
The Message-Passing Model

Values for SW1, SW2 (set by ISO):

```
+-----------------+-----------------+-----------------+
| Response Code   | Process Completed| Process Interrupted|
| (SW1, SW2)      |                 |                  |
|                 | Process Completed| Process Interrupted|
|                 | Normal Processing| Execution Error  |
|                 | Warning          | Checking Error   |
|                 | 61,XX or 90,00   | 64,XX or 65,XX   |
|                 | 62,XX or 63,XX   | 67,XX to 6F,XX   |
```
3. Processing APDUs

- Every time there is an incoming APDU for a selected applet:
  - The JCRE invokes the applet's `process()` method
  - The incoming APDU is passed as an argument
  - The applet must:
    - parse the command APDU
    - process the data
    - generate a response APDU
    - and return control to the JCRE
The Java Card RMI (JCRMI)

- The second communication model relies on a subset of the J2SE RMI distributed-object model
  - a server application creates and makes accessible remote objects
  - a client application obtains remote references to remote objects, and then invokes remote methods on them.

- In JCRMI, the Java Card applet is the server, and the host application the client
## Summary of Java Card Language Limitations

<table>
<thead>
<tr>
<th>Language Features</th>
<th>Dynamic class loading, security manager (<code>java.lang.SecurityManager</code>), threads, object cloning, and certain aspects of package access control are not supported.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keywords</td>
<td>native, synchronized, transient, volatile, strictfp are not supported.</td>
</tr>
<tr>
<td>Types</td>
<td>There is no support for <code>char</code>, <code>double</code>, <code>float</code>, and <code>long</code>, or for multidimensional arrays. Support for <code>int</code> is optional.</td>
</tr>
<tr>
<td>Classes and Interfaces</td>
<td>The Java core API classes and interfaces (<code>java.io</code>, <code>java.lang</code>, <code>java.util</code>) are unsupported except for <code>Object</code> and <code>Throwable</code>, and most methods of <code>Object</code> and <code>Throwable</code> are not available.</td>
</tr>
<tr>
<td>Exceptions</td>
<td>Some <code>Exception</code> and <code>Error</code> subclasses are omitted because the exceptions and errors they encapsulate cannot arise in the Java Card platform.</td>
</tr>
</tbody>
</table>
Questions?
References:

- http://developers.sun.com/techtopics/mobility/javacard/articles/javacard1/
- http://www.utoronto.ca/tcard/what.html
## Summary of Java Card Language Limitations

<table>
<thead>
<tr>
<th></th>
<th>Maximum Data Capacity</th>
<th>Processing Power</th>
<th>Cost of Card</th>
<th>Cost of Reader and Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnetic Stripe Cards</td>
<td>140 bytes</td>
<td>None</td>
<td>$0.20 - $0.75</td>
<td>$750</td>
</tr>
<tr>
<td>Integrated Circuit Memory Cards</td>
<td>1 Kbyte</td>
<td>None</td>
<td>$1 - $2.50</td>
<td>$500</td>
</tr>
<tr>
<td>Integrated Circuit Processor Cards</td>
<td>8 Kbytes</td>
<td>8-bit cpu, moving to 16- and 32-bit</td>
<td>$7-$15</td>
<td>$500</td>
</tr>
<tr>
<td>Optical Memory Cards</td>
<td>4.9 Mbytes</td>
<td>None</td>
<td>$7 - $12</td>
<td>$3,500 - $4,000</td>
</tr>
</tbody>
</table>

Smart Card Types and Their Capabilities

- Integrated Circuit (IC) Microprocessor Cards:
  - offer greater memory storage and security of data than a traditional mag stripe card
  - Chip cards also can process data on the card.

- Integrated Circuit (IC) Memory Cards.
  - IC memory cards can hold up to 1-4 KB of data
  - but have no processor
  - they are dependent on the card reader for their processing

- Optical Memory Cards:
  - look like a card with a piece of a CD glued on top store up to 4 MB of data