Un Survol de Java Card

Ewen Denney

Projet Lande
IRISA, Rennes

12 février 1999

Cartes à Puces

Cartes à puces

mémoire, E/S avec CAD, processeur

Types

avec contact, sans contact, hybride

Applications

contrôle d'accès, voyage, portefeuille électronique, cartes de santé, ...

Avantages

sûreté, confidentialité, grande capacité mémoire, multi-application, compatibilité avec périphériques, peut évoluer

Programmation des cartes

Limitations physiques

8 bit, 3-4 MHz, 24K ROM, 16K EEPROM, 512b RAM

Applications aux cartes à puces

communication avec lecteurs, PINs, installation, initialisation, personnalisation, muting, tearing

Standards et spécifications

Vertical

EMV, GSM, UMTS, IATA

Horizontal

ISO 7816, PC/SC

Caractéristiques particulières

- fichiers orientés-objet
- noyau de système opératoire + applettes
- APIs
- chargeurs des classe
- sécurité d'exécution
- partage des objets
- atomicité des transactions

La famille Java

- Java, Java Card, Personal Java, Embedded Java, JINI
- Java Wallet, Open Card
- ‘Java Card Technology’ = JCVM + API + langage

Java

Orienté-objet

classes, objets, champs, methodes (statique et dynamique), héritage, interfaces

Packages

`package`, `import`, noms complètement qualifiés

Exceptions

`throw` *`exception`*

`try {` *`command`* `} catch` *`exception`* `{` *`command`* `} finally {` *`command`* `}`

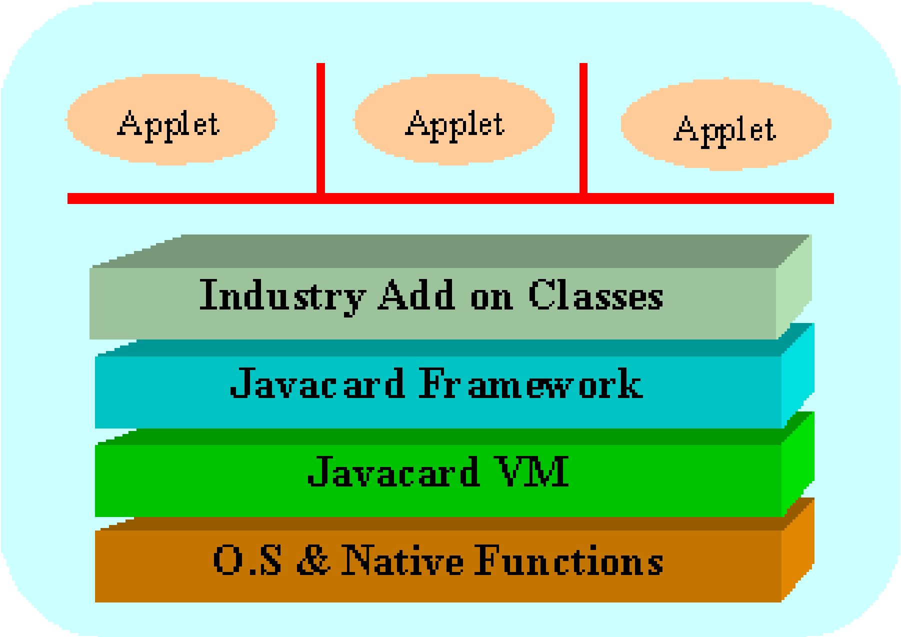
Attributs de protection

`static`, `final`, `public`, `private`, `protected` + ‘package’

Spécificités de Java Card

- pas de threads, clonage des objets, types ‘grands’ (float, double, …)
- entiers sont facultatifs
- pas de chargement dynamique des classes
- objets persistants/transiants
- sécurité au niveau de la machine virtuelle
- partage des objets
- atomicité des transaction

$$\text{source code} \xrightarrow{compilateur} \text{class file} \xrightarrow{convertisseur} \text{format optimisé}$$



Java Card API 2.0

`java.lang`

objets, exceptions

`javacard.framework`

AIDs, APDUs, applettes, ISO

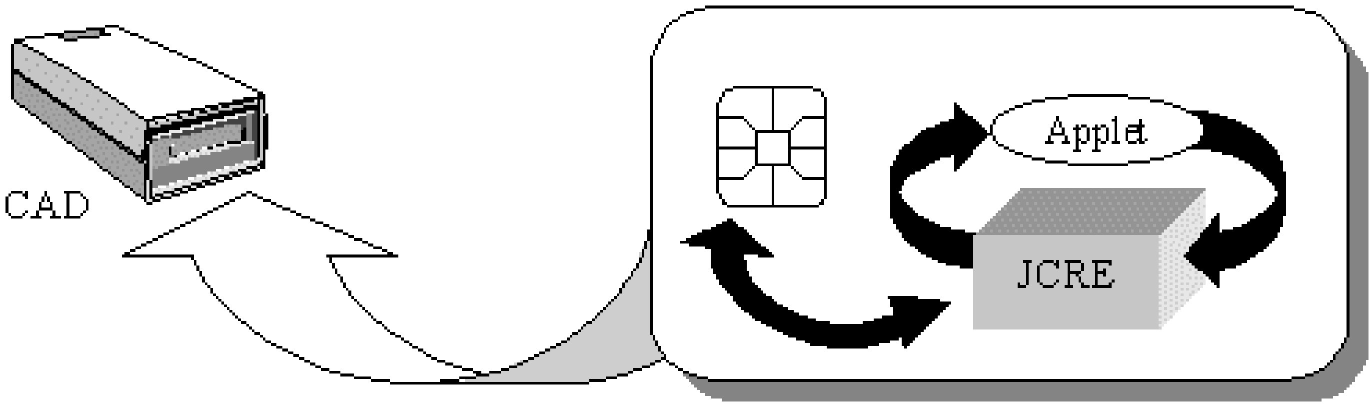
`javacardx.framework`

fichiers orientés-objet

`javacardx.framework.crypto`

`javacardx.framework.cryptoEnc`

Applettes



Applette Une petite application

Communication Commande-réponse en forme de APDUs

Cycle de vie d'une carte

1. Masking
2. Initialisation
3. Personnalisation
4. Exécution des applettes:

   installation

   `install(APDU apdu)`

   sélection

   `select()`

   processing

   `process(APDU apdu)`

   désélection

   `deselect()`

Isolement et partage

- applettes ne peuvent pas avoir accès aux membres sauf une interface explicite
- *firewalls* cloisonnent les applettes dans *contextes*
- contexte actuel; pile des contextes; peut changer
- pour accès sécurisé entre contextes:
  - global arrays
  - JCRE est privilégié
  - shareable interfaces

Shareable interfaces

Portefeuille électronique

```
package bank.purse;


  import javacard.framework.*;
  import javacardx.framework.*;

  public class Wallet extends Applet {

    /* constants declaration */

    final static byte Wallet_CLA =(byte)0xB0;
    final static byte Deposit = (byte) 0x10;
    final static byte Debit = (byte) 0x20;
    final static byte Balance = (byte) 0x30;
    final static byte Validate = (byte) 0x40;
    final static byte PinTryLimit =(byte)0x03;
    final static byte MaxPinSize =(byte)0x04;

    OwnerPIN pin;
    byte balance;
    byte buffer[];
```

```
private Wallet() {

  pin = new OwnerPIN(PinTryLimit, MaxPinSize);
  balance = 0;
  register();

} // end of the constructor


public static void install(APDU apdu){

  new Wallet();

} // end of install method

  public boolean select() {

  pin.reset();
  return true;

}// end of select method
```

```
  public void process(APDU apdu) {

    buffer = apdu.getBuffer();

    if (buffer[ISO.OFFSET_CLA] !== Wallet_CLA)
      ISOException.throwIt
           (ISO.SW_CLA_NOT_SUPPORTED);

    switch (buffer[ISO.OFFSET_INS]) {
      case Balance:   getBalance(apdu); return;
      case Debit:   debit(apdu); return;
      case Deposit:   deposit(apdu);return;
      case Validate:    validate(apdu);return
      default:   ISOException.throwIt
             (ISO.SW_INS_NOT_SUPPORTED);

    }

}   // end of process method
```

```
  private void debit(APDU apdu) {

    if ( ! pin.isValidated() )
      ISOException.throwIt(ISO.SW_PIN_REQUIRED);
    byte numBytes = (byte)(buffer[ISO.OFFSET_LC]);
    byte byteRead = (byte)(apdu.setIncomingAndReceive());
    if (byteRead != 1)
      ISOException.throwIt(ISO.SW_WRONG_LENGTH);
    if ( ( balance - buffer[ISO.OFFSET_CDATA]) < 0 )
      ISOException.throwIt(SW_NEGATIVE_BALANCE);
    balance = (byte)
        (balance - buffer[ISO.OFFSET_CDATA]);

} // end of debit method
```

Questions intéressantes

- prouvabilité est possible et désirable
- correction de bytecode
- correction des compilateurs et convertisseurs
- correction par rapport aux standards et APIs
- châine de sécurité
- modelisation du partage et atomicité
- modularité de la sémantique: sous-langages, APIs, versions, désapprouvation