La Carte à Microprocesseur

Un système embarqué en plein essor

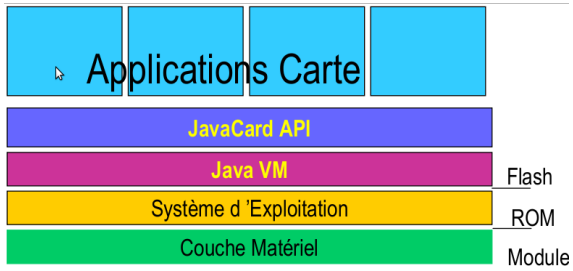
Tegawendé F. Bissyandé
tegawende.bissyande@fasolabs.org

Cours préparé pour
L'Université Ouaga I Pr. Joseph Ki-Zerbo(UFR SEA)

April 7, 2015

FasoLabs.org

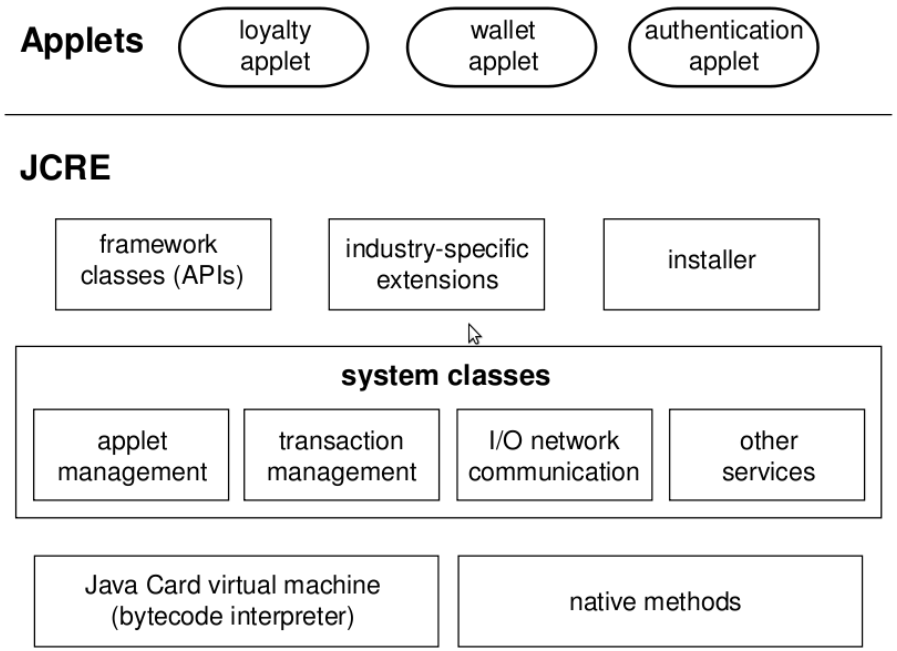
La Java Card



- Pas de chargement dynamique de classe
- Allocation dynamique d'objets supportée mais
  - pas de garbage collection
  - pas de désallocation explicite non plus :(
  - ...
- Quelques types de base (`byte`, `int`, `boolean`) – pas de `char` (ni de classe `String`), `double`, `float`, `long`
- Objets supportés / Mécanisme d'héritage supporté / Pas de threads
- Securité : Notion de paquetages & modifieurs `public`, `private`, `protected`

FasoLabs.org

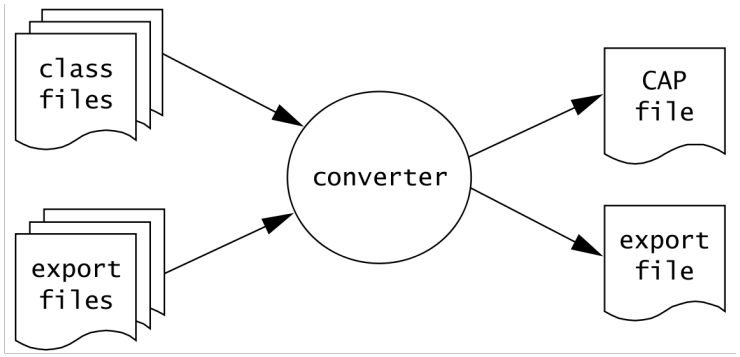
Architecture de la Java Card

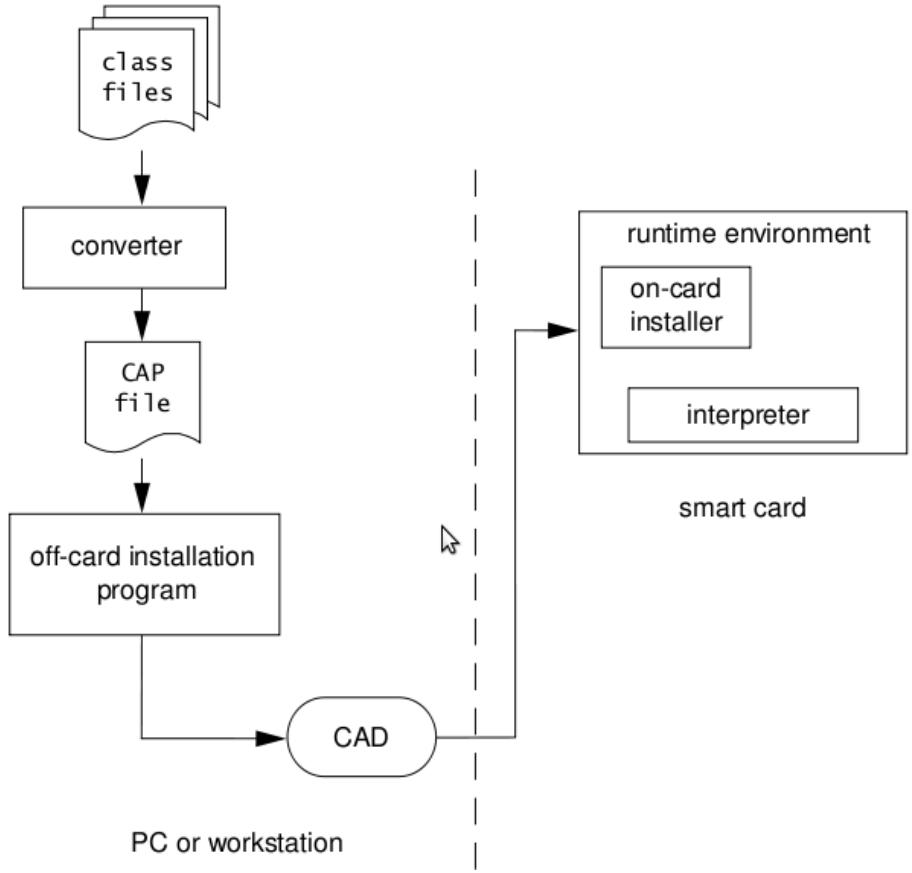


- JCVM : Java Card Virtual Machine
  - allocation mémoire
  - exécution du bytecode
- JCRE : Java card runtime environment
  - chargée à l'usine
  - copie les données de EEPROM et ROM vers RAM
- native methods
  - communication bas niveaux
  - support cryptographique

FasoLabs.org

Retour sur la compilation





Construction d'applications Java Card

Une application carte

- Code dans la carte (app serveur = applet Java card)
- Code dans le terminal (app cliente)

Construction d'une application Java card

- Construction de l'applet (implementation des services)
- Installation de l'applet dans les cartes (Initialisation des services)
- Construction de l'application cliente (Invocation des services)

FasoLabs.org

Construction d'applications Java Card

Une application carte

- Code dans la carte (app serveur = applet Java card)
- Code dans le terminal (app cliente)

Construction d'une application Java card

- Construction de l'applet (implementation des services)
- Installation de l'applet dans les cartes (Initialisation des services)
- Construction de l'application cliente (Invocation des services)

FasoLabs.org

Programming....

- Installation de l'applet Java Card
  - Compilation, conversion et chargement sécurisé de l'applet dans les cartes (Java Card IDE)
  - Appel à la méthode `install(APDU apdu)` des applets (non standardisé)
    - L'APDU contient les paramètres d'initialisation de l'applet

FasoLabs.org

Programming....

- Construction de l'applet Java Card
  - Implémentation des classes de l'applet avec l'API Java Card
  - Définition des APDUs de commande traités par l'applet et des APDUs de réponse renvoyés par l'applet (données ou erreurs)
  - → implémentation de la méthode `process(APDU apdu)`
  - Le JCRE fournit l'environnement d'exécution et la couche de communication

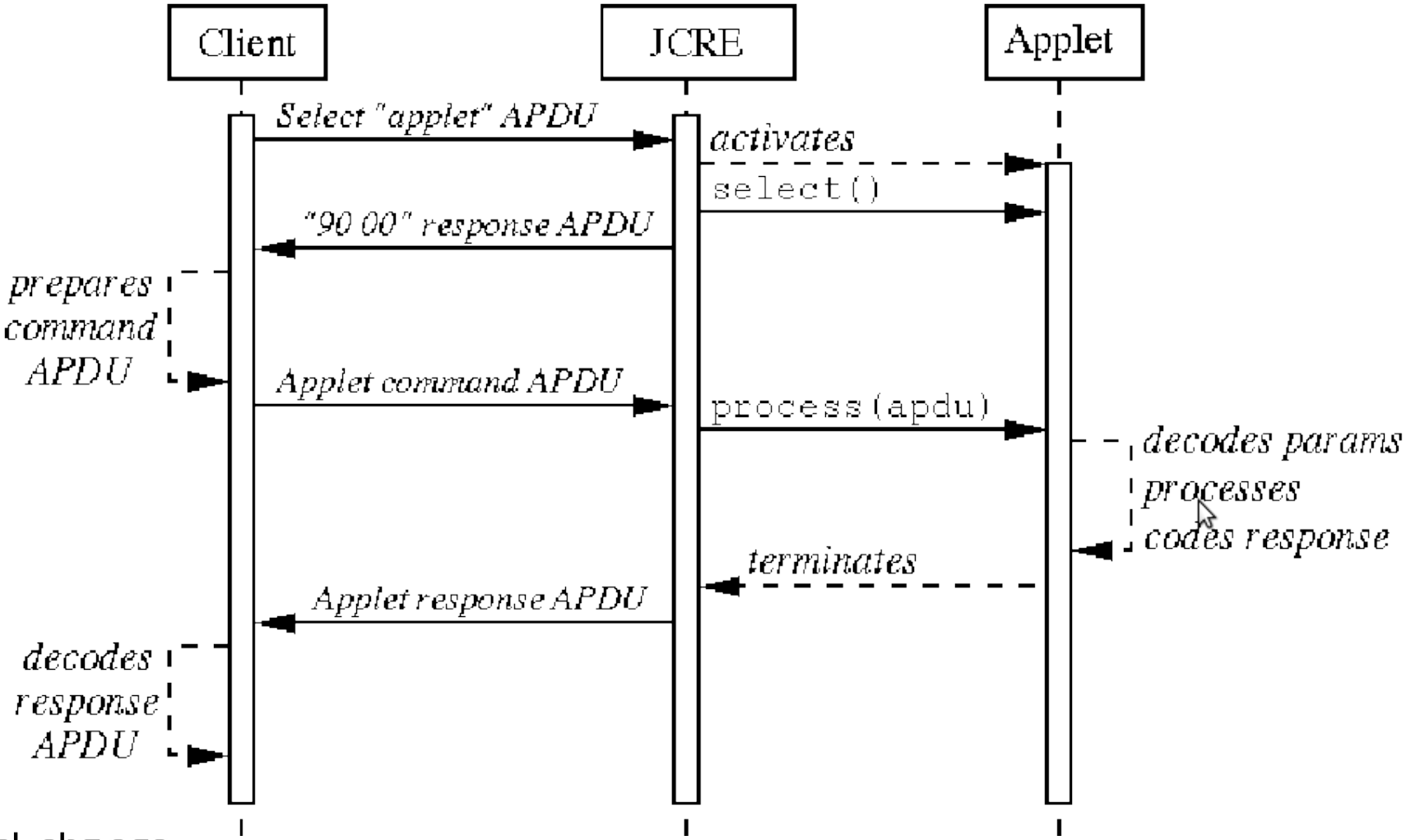
FasoLabs.org

Programming....

- Construction de l'application terminal
  - Implémentation des classes du terminal (avec JDK)
  - Communication avec le serveur (applet carte)
    - Établissement de la liaison : envoi d'un APDU de sélection avec l'AID de l'applet (standardisé)
    - Invocation de services de l'applet :
      - codage et envoi d'APDUs de commande conformes à ceux traités par l'applet
      - réception et décodage des APDUs de réponse retournés par l'applet
  - Pas d'API standard de communication avec la carte

FasoLabs.org

Programming....



FasoLabs.org

Programming....

- Un simple Compteur
  - Carte de fidélité, Porte Monnaie Electronique, ...
- APDUs traités par l'applet :
  - `int lire()`
    - Commande : `AA 01 XX XX 00 04`
    - Réponse : `RV3 RV2 RV1 RV0 90 00`
  - `int incrementer( int )`
    - Commande : `AA 02 XX XX 04 AM3 AM2 AM1 AM0 04`
    - Réponse : `RV3 RV2 RV1 RV0 90 00`
  - `int decrementer( int )`
    - Commande : `AA 03 XX XX 04 AM3 AM2 AM1 AM0 04`
    - Réponse : `RV3 RV2 RV1 RV0 90 00`

FasoLabs.org

Programming....

```
package org.carte.compteur ;

import javacard.framework.* ;

public class Compteur extends Applet {
  private int valeur;

  public Compteur() { valeur = 0; register(); }
  public static void install( APDU apdu ) { new Compteur(); }

  public void process( APDU apdu ) {
    byte[] buffer = apdu.getBuffer();
    if ( buffer[ISO.OFFSET_CLA] != 0xAA )
      ISOException.throwIt(ISO.SW_CLA_NOT_SUPPORTED);
    switch ( buffer[ISO.OFFSET_INS] ) {
      case 0x01: ... // Opération de lecture
      case 0x02: ... // Opération d'incrémentation
      case 0x03: ... // Opération de décrémentation
      default:
        ISOException.throwIt(ISO.SW_INS_NOT_SUPPORTED);
    }
  }
}
```

FasoLabs.org

Programming....

```
case 0x03: // Opération de décrémentation
{
  // Réception des donnnées
  byte octetsLus = apdu.setIncomingAndReceive();
  if ( octetsLus != 4 )
    ISOException.throwIt(ISO.SW_WRONG_LENGTH);
  int montant = (buffer[ISO.OFFSET_CDATA]<<24) |
    (buffer[ISO.OFFSET_CDATA+1]<<16) |
    (buffer[ISO.OFFSET_CDATA+2]<<8) |
    buffer[ISO.OFFSET_CDATA+3];
  // Traitement
  if ( montant<0 || valeur-montant<0 )
    ISOException.throwIt((short)0x6910);
  valeur = valeur - montant;
  // Envoie de la réponse
  buffer[0] = (byte)(valeur>>24);
  buffer[1] = (byte)(valeur>>16);
  buffer[2] = (byte)(valeur>>8);
  buffer[3] = (byte)(valeur);
  apdu.setOutgoingAndSend((short)0, (short)4);
  return;
}
```


Programming....

```
package com.banque ;

import javacard.framework.*;

public class Pme extends Applet {
  final static byte Pme_CLA         = (byte)0xB0;
  final static byte Crediter_INS    = (byte)0x10;
  final static byte Debiter_INS     = (byte)0x20;
  final static byte Lire_INS        = (byte)0x30;
  final static byte Valider_INS     = (byte)0x40;
  final static byte MaxEssai_PIN    = (byte)0x03;
  final static byte MaxLg_PIN       = (byte)0x08;
  final static short BalanceNegative_SW = (short)0x6910;

  OwnerPin pin;
  byte balance;
  byte[] buffer;

  private Pme() {
    pin = new OwnerPIN(MaxEssai_PIN, MaxLg_PIN);
    balance = 0;
    register() ;
  }
```


Programming....

```
public static void install(byte[] bArray,
                           short bOffset, byte bLength) {
   Pme p=new Pme();
   pin.updateAndUnblock(bArray, bOffset, bLength);
}

public boolean select() { pin.reset(); return true; }

public void process( APDU apdu ) {
   buffer = apdu.getBuffer();
   if ( buffer[ISO.OFFSET_CLA] != Pme_CLA )
     ISOException.throwIt(ISO.SW_CLA_NOT_SUPPORTED);
   switch ( buffer[ISO.OFFSET_INS] ) {
     case Crediter_INS : crediter(apdu); return;
     case Debiter_INS : debiter(apdu); return;
     case Lire_INS : lire(apdu); return;
     case Valider_INS : valider(apdu); return;
     default:
       ISOEXception.throwIt(ISO.SW_INS_NOT_SUPPORTED);
   }
```

FasoLabs.org

Programming....

```
// Réception de données
private void crediter( APDU apdu ) {
  if ( !pin.isValidated() )
    ISOException.throwIt(ISO.SW_PIN_RIQUIRED);
  byte octetsLus = apdu.setIncomingAndReceive();
  if ( octetsLus != 1 )
    ISOException.throwIt(ISO.SW_WRONG_LENGTH);
  balance = (byte)(balance + buffer[ISO.OFFSET_CDATA]);
}

// Réception de données
private void debiter( APDU apdu ) {
  if ( !pin.isValidated() )
    ISOException.throwIt(ISO.SW_PIN_RIQUIRED);
  byte octetsLus = apdu.setI..comingAndReceive();
  if ( octetsLus != 1 )
    ISOException.throwIt(ISO.SW_WRONG_LENGTH);
  if ( (balance - buffer[ISO.OFFSET_CDATA]) < 0 )
    ISOException.throwIt(BalanceNegative_SW);
  balance = (byte)(balance - buffer[ISO.OFFSET_CDATA]);
}
```

Programming....

```
    // Émission de données
    private void lire( APDU apdu ) {
      if ( !pin.isValidated() )
        ISOException.throwIt(ISO.SW_PIN_RIQUIRED);
      apdu.setOutgoing();
      apdu.setOutgoingLength((byte)1);
      buffer[0] = balance;
      apdu.sendBytes((short)0, (short)1) ;
    }

    // Manipulation du code secret
    private void valider( APDU apdu ) {
      byte octetsLus = apdu.setIncomingAndReceive();
      pin.check(buffer, ISO.OFFSET_CDATA, octetsLus);
    }

}
```

FasoLabs.org

Programming....

```
private DESKey myDESKey;

public static void install(byte[] bArray,
                           short bOffset, byte bLength) {
    new Encryption ();
    pin.updateAndUnblock(bArray, bOffset, bLength);
 }

  public boolean select() { pin.reset(); return true; }

  public void process( APDU apdu ) {
    buffer = apdu.getBuffer();
    if ( buffer[ISO.OFFSET_CLA] !=0x00 )
      ISOException.throwIt(ISO.SW_CLA_NOT_SUPPORTED);
    switch ( buffer[ISO.OFFSET_INS] ) {
      case ENCRYPT_INS : encrypt(apdu); return;
      case PINCHECK_INS : pinCheck(apdu); return;
     default:
       ISOEXception.throwIt(ISO.SW_INS_NOT_SUPPORTED);
    }
```

FasoLabs.org

Programming....

- Classe dérivant de `javacard.framework.Applet`
- Une applet carte
  est un programme serveur de la Java Card
  - APDU de sélection depuis le terminal (select)
  - Sélection par AID (chaque applet doit avoir un AID unique)
    - AID
      - 5 octets identifiant le propriétaire
      - 0-11 octets dépendant du propriétaire

FasoLabs.org

Programming....

- Classe dérivant de `javacard.framework.Applet`
- Une applet carte
  est un programme serveur de la Java Card
  - APDU de sélection depuis le terminal (select)
  - Sélection par AID (chaque applet doit avoir un AID unique)
    - AID
      - 5 octets identifiant le propriétaire
      - 0-11 octets dépendant du propriétaire

FasoLabs.org

Programming....

- Cycle de vie : Méthodes appelées par JCRE
  - `static void install(bArray,bOffset,bLength)`
    - Crée une instance de la classe avec les paramètres passés dans bArray
    - Puis l'enregistre (register()) auprès du JCRE
  - `boolean select()`
    - Appelé à la sélection
    - peut retourner false si l'initialisation est incomplète (liaison impossible vers des objets partagés, ...)
  - `void deselect()`
    - Appelé à la désélection
  - `void process(APDU apdu)`
    - Méthodes appelées par JCRE

FasoLabs.org

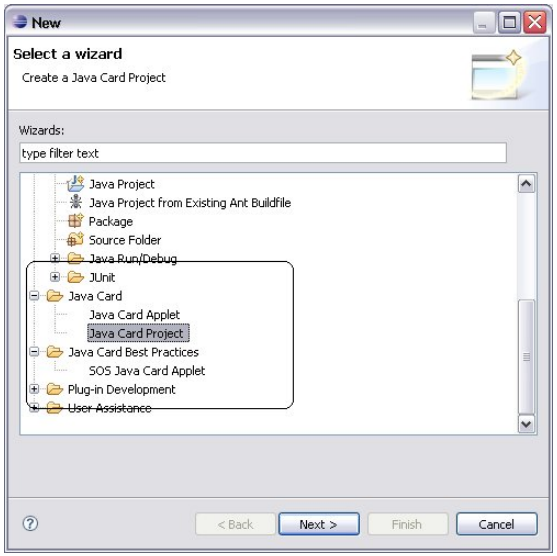
Version 2.2 de l'API de Java Card

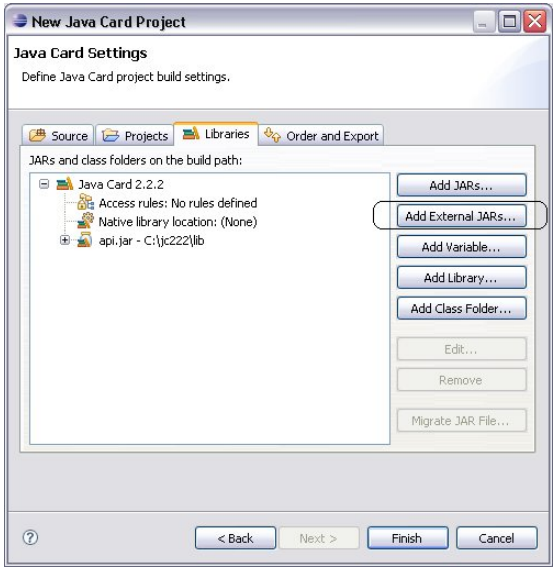
Package Summary	
Packages	
java.io	A subset of the java.io package in the standard Java programming language.
java.lang	Provides classes that are fundamental to the design of the Java Card technology subset of the Java programming language.
java.rmi	The java.rmi package defines the Remote interface which 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 applets.
javacard.framework.service	Provides a service framework of classes and interfaces that allow a Java Card 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 Java Card.
javacardx.crypto	Extension package that contains functionality, which may be subject to export controls, for implementing a security and cryptography framework on Java Card.

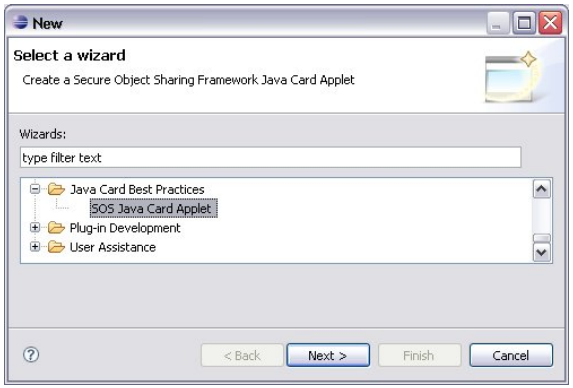
FasoLabs.org

EclipseJCDE – la vie devient facile

Rendez-vous au plus à cet url pour les windowsiens :(– `http://eclipse-jcde.sourceforge.net/`







FasoLabs.org