# La technologie Java Card

*Samia Bouzefrane*

Maître de Conférences

CEDRIC –CNAM

samia.bouzefrane@cnam.fr
http://cedric.cnam.fr/~bouzefra

# La technologie Java Card : introduction et principe



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Java Card - Introduction

- **Besoin des systèmes « programmables »**

- **Recherche de solution « évolutive » (dépasser la ROM)**

- **Les applications :**
  - ✓ Longues à développer
  - ✓Statiques dans leurs fonctions

- **Des tentatives**
  - ✓ 1ère version: octobre 1996, démarrage et produit réel
    en 1998, une réalité industrielle à partir de 2000. En 2004, le nombre de Java Cards vendus a atteint le milliard.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Étapes du développement de l'industrie

➢**La carte à microprocesseur et les grandes étapes de développement de la technologie :**

✓ Les pionniers (1975-1985): premières idées
(les bases technologiques sont établies)

✓1985-1995: la technologie est améliorée
- marchés et déploiements importants: CB, GSM
- limites : besoin de davantage de flexibilité

✓1995-2005 : explosion du marché, nouveau paradigme avec
- les cartes évolutives basées sur Java Card

✓2006: 1,2 billions de téléphones mobiles utilisant des cartes SIM/Java Card
1,65 billions cartes à puce/ Java Card (source site de Sun )

✓2005-???: la carte devient un élément du réseau
-Les SCWS (Smart Card Web Server)
- .Net


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# L'arrivée de la technologie Java Card

➢**Novembre 1996, première proposition d'utilisation de Java pour les cartes est faite par une équipe de Schlumberger (Austin)**

- ✓ proposition de Java Card API permettant la programmation en Java de la carte
- ✓C'est la Java Card 1.0

➢ **Bull, Gemplus et Schlumberger créent le Java Card Forum**

- ✓ le JCF discute et propose à Sun des spécifications

➢**Novembre 1997, publication de la spécification Java Card 2.0**

- ✓Gemplus démontre en oct./nov. CASCADE, le premier chip RISC 32 bit (ARM 7) avec de la mémoire Flash, « une » implémentation de la Java Card 2.0 et des DMIs (Direct Method Invocation), etc.

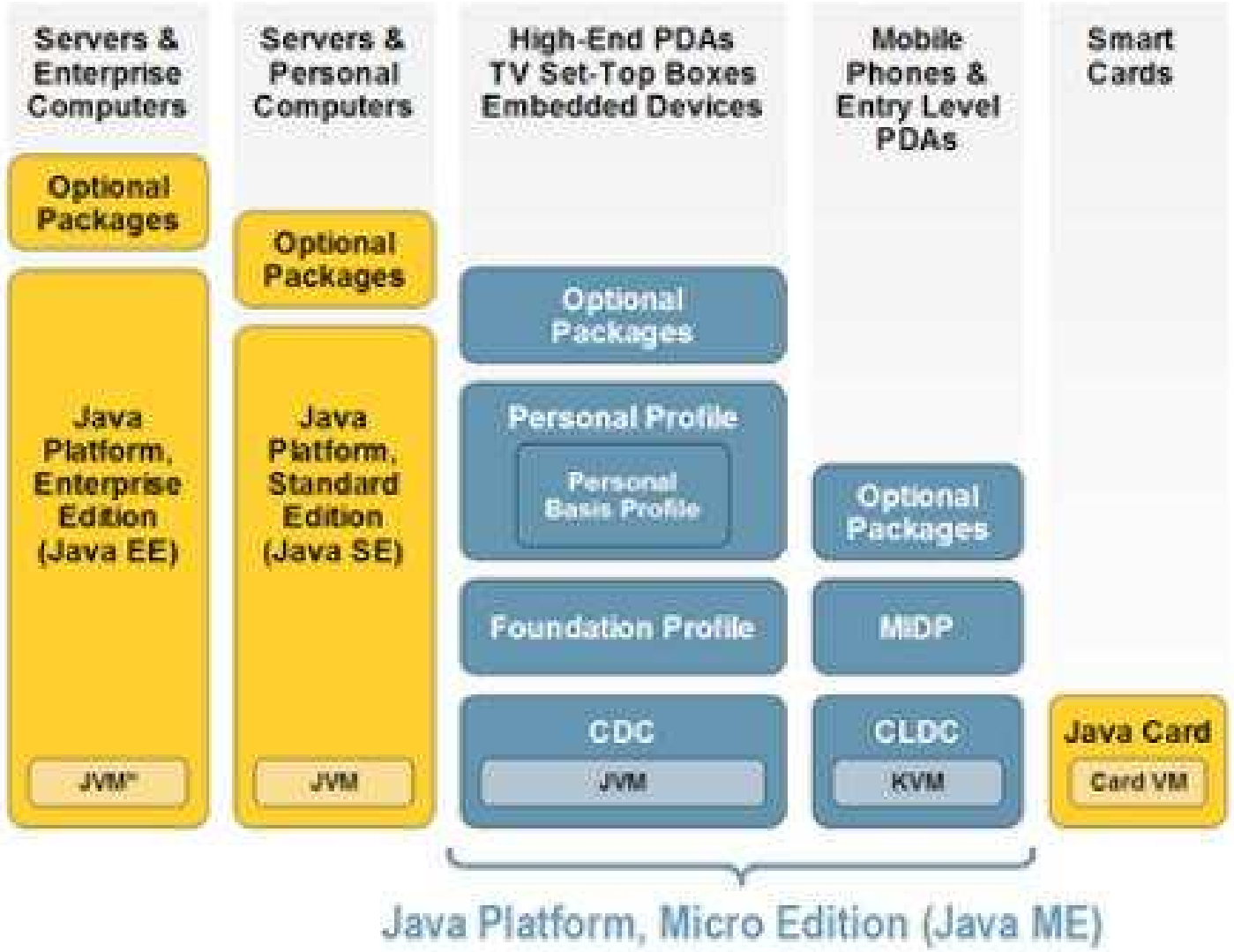

samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Les évolutions de la 2.x

- **La version 2.0 de la Java Card Specification introduit:**
  - ✓ un environnement d'exécution
  - ✓La possibilité d'écrire des applets avec une approche orientée objet (même si le format de chargement n'était pas encore spécifié)
- **Mars 1999, la version 2.1 qui contient 3 parties est publiée:**
  - ✓ Java Card API Specification
  - ✓Java Card Runtime Environment Specification
  - ✓Java Card Virtual Machine Specification


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Un élément de la technologie Java






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# A propos du modèle de licence/1

- **La spécification est disponible sur :**
  - ✓ http://java.sun.com/products/javacard/
- **Vendre des cartes (avec ou sans logo) et afficher une compatibilité avec la technologie implique d'être licencié Java Card Technology**
- **Ce qui donne accès à :**
  - ✓Une implémentation de référence
  - ✓La suite de tests de compatibilité
  - ✓Du support spécifique


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# A propos du modèle de licence/2

➢**Java Authorized Licensees of Java Card Technology**

✓ the companies listed below licensed Java Card technology from the Sun MicroSystems. Only Java Card licensees can ship products that bear the « Java Powered » logo and claim compatibility with the Java Card Platform specification and Java Card TCK.

✓ ARM, Aspects, CCL/ITRL, Fujitsu, Gemplus, SAGEM, Oberthur Card Systems, Trusted Logic, etc.

Source : http://java.sun.com/products/javacard/licensees.html



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Le Java Card Forum

- **Association regroupant les fabricants de silicium, les encarteurs et les clients**
  - ✓ Promouvoir la technologie Java Card
  - ✓Définir des choix technologiques puis les proposer à Sun qui en fait le « standard ».
- JCF : http://www.javacardforum.org


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

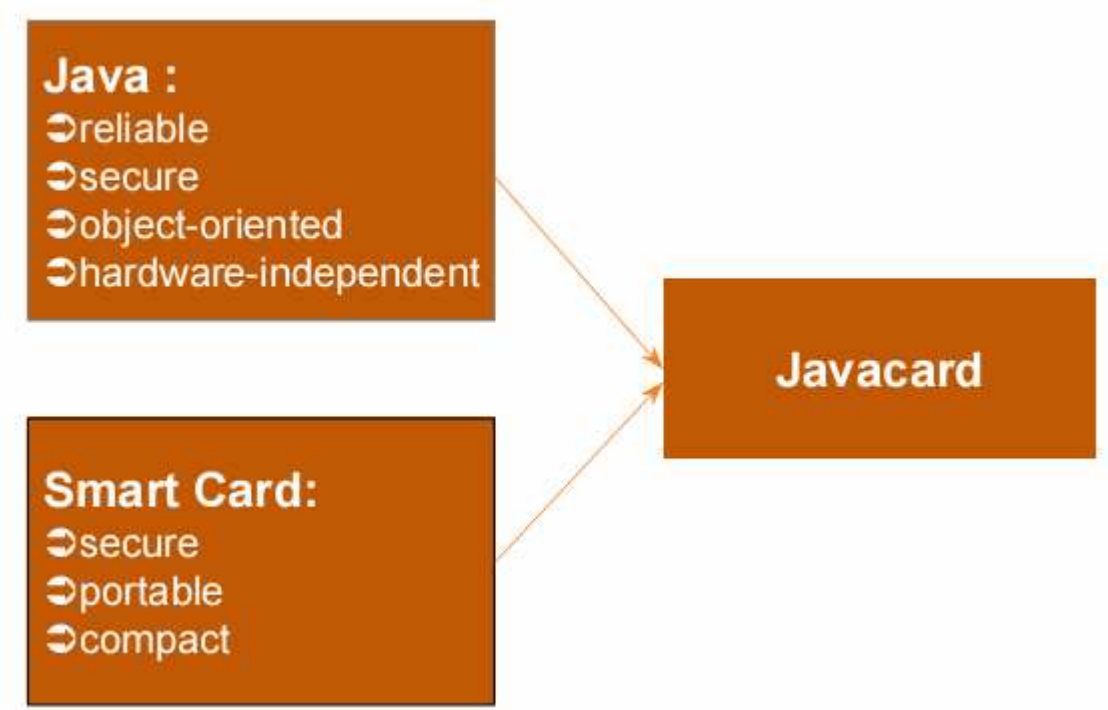
# Une plate-forme Java Card (résumé)

- est une carte à puce
- avec une machine virtuelle Java
- capable d'exécuter des applications écrites en Java

- Les plateformes Java Card sont normalisées par Sun et Java Card Forum
- Java est le langage de programmation le plus utilisé dans les applications dédiées à la carte à puce


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Java Card = Java + smart Card






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Carte à puce standard

- Application, OS et hardware sont liés
- L'application est développée uniquement par le propriétaire de l'OS
- L'application est développée dans un langage de bas niveau (C, Assembleur)
- Cycle de développement = 5 mois
- Pas de vraie multi-application (données uniquement)


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Plate-forme Java Card

- Application, OS et hardware sont indépendants
- L'application est développée par n'importe quel programmeur Java
- L'application est développée dans un langage standard (de haut niveau)
- Cycle de développement = 2 mois
- Carte multi-application (données + code)


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Avantages de la technologie Java Card

- Développement facile
- Interopérabilité des applets (utilisation sur des plateformes différentes)
- Sécurité (sécurité du langage, optimisation, etc.)
- Multi-application
- Dynamicité
- Ouverture et compatibilité (rajout et maj d'applications)
- Capacité de post-personnalisation

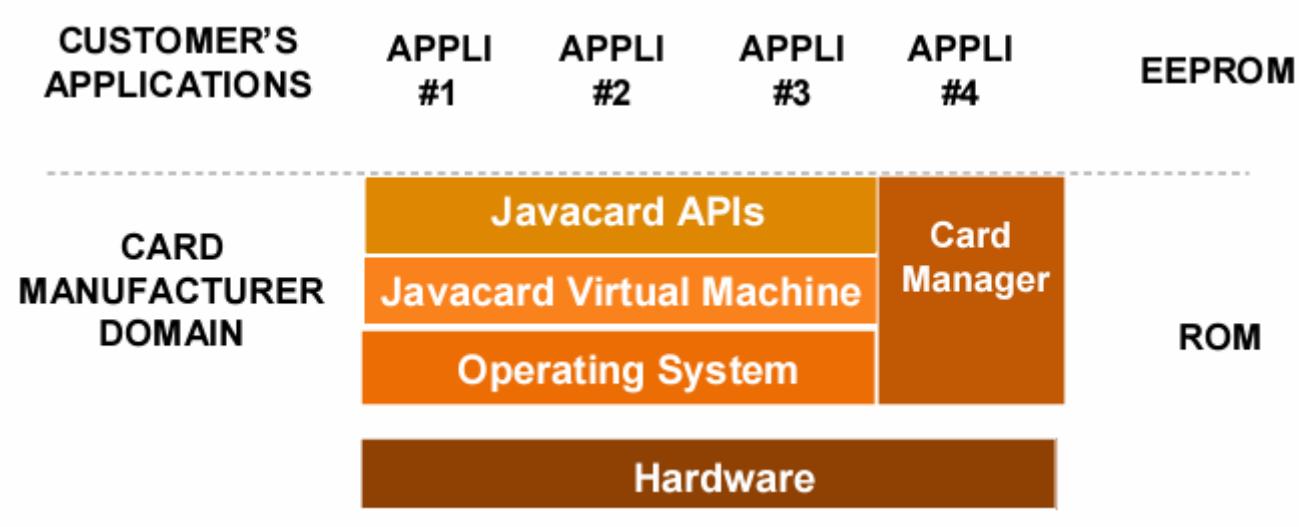

samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Le langage Java Card



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Acteurs d'une Java Card






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Caractéristiques d'une Java Card

➢**Prise en compte des architectures matérielles des cartes dont les tailles sont très réduites : moins de 1Ko de RAM, 24-28 Ko de ROM et 8 à 16 Ko NVM (EEPROM).**

➢ **Pour intégrer la technologie Java dans une carte, les choix sont les suivants :**

- ✓Réduire les fonctionnalités du langage
- ✓Minima requis pour faire tourner un programme Java Card sont :
  24 ko de ROM, 16 ko d'EEPROM et 1 ko de RAM.
- ✓Distribuer le modèle de la JVM entre le « on Card » et le « off Card »

➢**Trois parties :**

- ✓Java Card API Specification
- ✓Java Card Runtime Environment Specification
- ✓Java Card Virtual Machine Specification


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Types supportés

| Type | Size | Supported in Javacard |
|---|---|---|
| boolean | - | yes |
| byte | 8-bit | yes |
| short | 16-bit | yes |
| int | 32-bit | yes (optional) |
| long | 64-bit | no |
| double | 64-bit | no |
| char | 16-bit | no |
| String | variable | no |
| float | 32-bit | no |



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Caractéristiques non supportées

- Pas de Threads
- Pas de chargement dynamique
- Pas de Garbage Collector (ramasse-miettes, jusqu'à la version 2.2)
- Pas de clonage
- Pas de tableaux à plusieurs dimensions


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Résumé des caractéristiques

| Caractéristiques supportées | Caractéristiques non supportées |
|---|---|
| boolean, byte, short | long, double, float, char, String |
| Tableau à une dimension | Tableau à plusieurs dimensions |
| Paquetage Java, classes, interface et exceptions | Threads, sérialisation |
| Héritage, méthode abstraite, surcharge et création d'objets (instantiation) | Chargement dynamique de classes |
| « int » est optionnel | Gestionnaire de sécurité |



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Mots clés

- **Mots clés supportés**

abstract, boolean, break, byte, case, catch, class, const, continue, default, do, else, extends, false, final,goto null, package, private, protected, public, return, static, super, switch, this, if, implements, import, instanceof, int, interface, new, null, package, private, protected, public, return, short, static, super, switch, this, throw, true, try, void, while.

- **Identificateurs non supportés**

char, double, float, long, native, synchronized, transient, threadsafe, volatile, finalize



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Caractéristiques spécifiques à Java Card

- Objets temporaires (APDU, Reset, Select)
- Atomicité
- Partage
- Gestion d'exceptions sur la carte (runtime, ISO)
- API particulière : Java Card 2.1.x et 2.2
- Méthodes spéciales pour l'installation d'applets, d'APDUs et de sélection


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Objets temporaires (transient)

- **Définition :**
  - ✓ **objets dont les champs sont effacés suite à un événement**

- **Caractéristiques**
  - ✓ **La valeur est effacée et non l'objet lui-même**
  - ✓ **localisés dans la RAM**
  - ✓ **utilisés pour les données temporaires, fréquemment modifiées**

- **Événements qui réinitialisent les objets temporaires**
  - ✓ **Reset, Select, Deselect.**


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Atomicité / Transaction

- **Définition :**
  - ✓ **une transaction est atomique si tous ses champs sont mis à jour ou pas du tout**

- **Caractéristiques**

  - ✓ **si une transaction ne se termine pas normalement (coupure de courant, carte retirée, etc.), les données sont remises à leurs valeurs initiales**

  - ✓ **évite la perte de données sensibles (ex. la balance dans le porte-monnaie)**

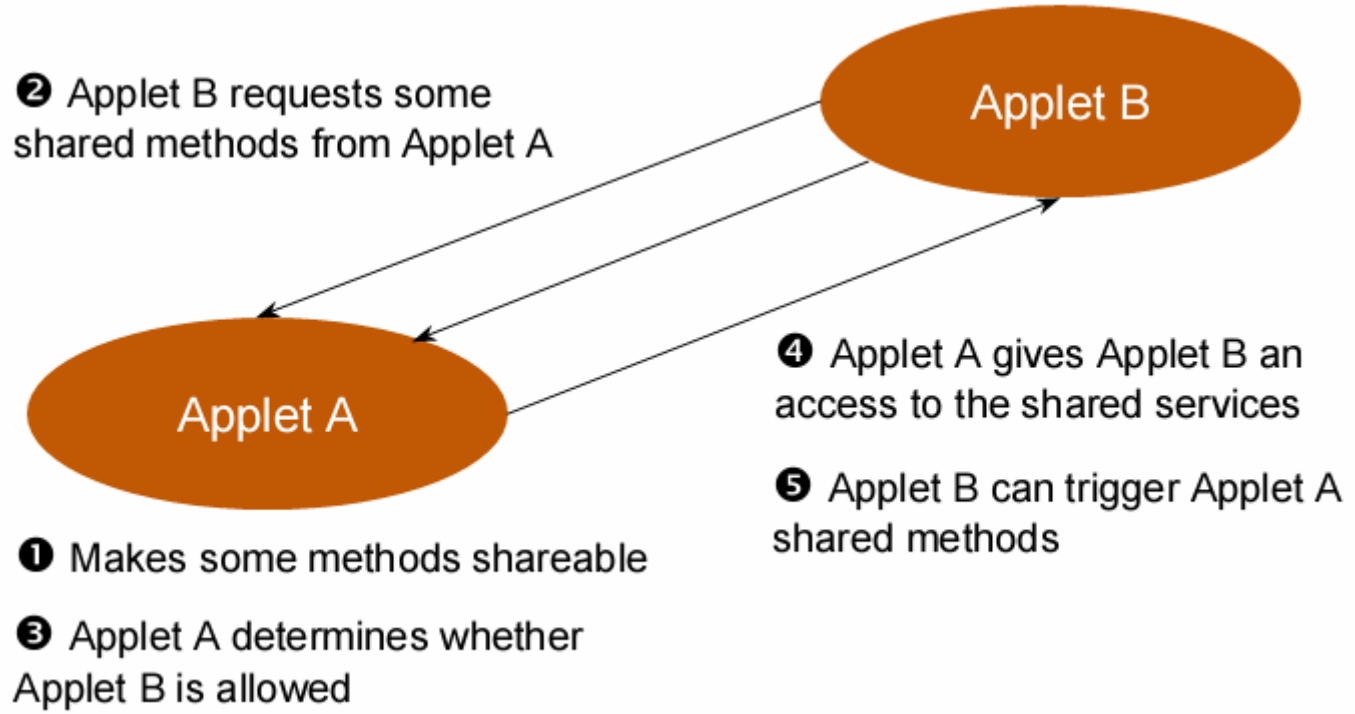
  - ✓ **le mode transactionnel peut être positionné ou non**

  - ✓ **gestion de l'atomicité via l'API**

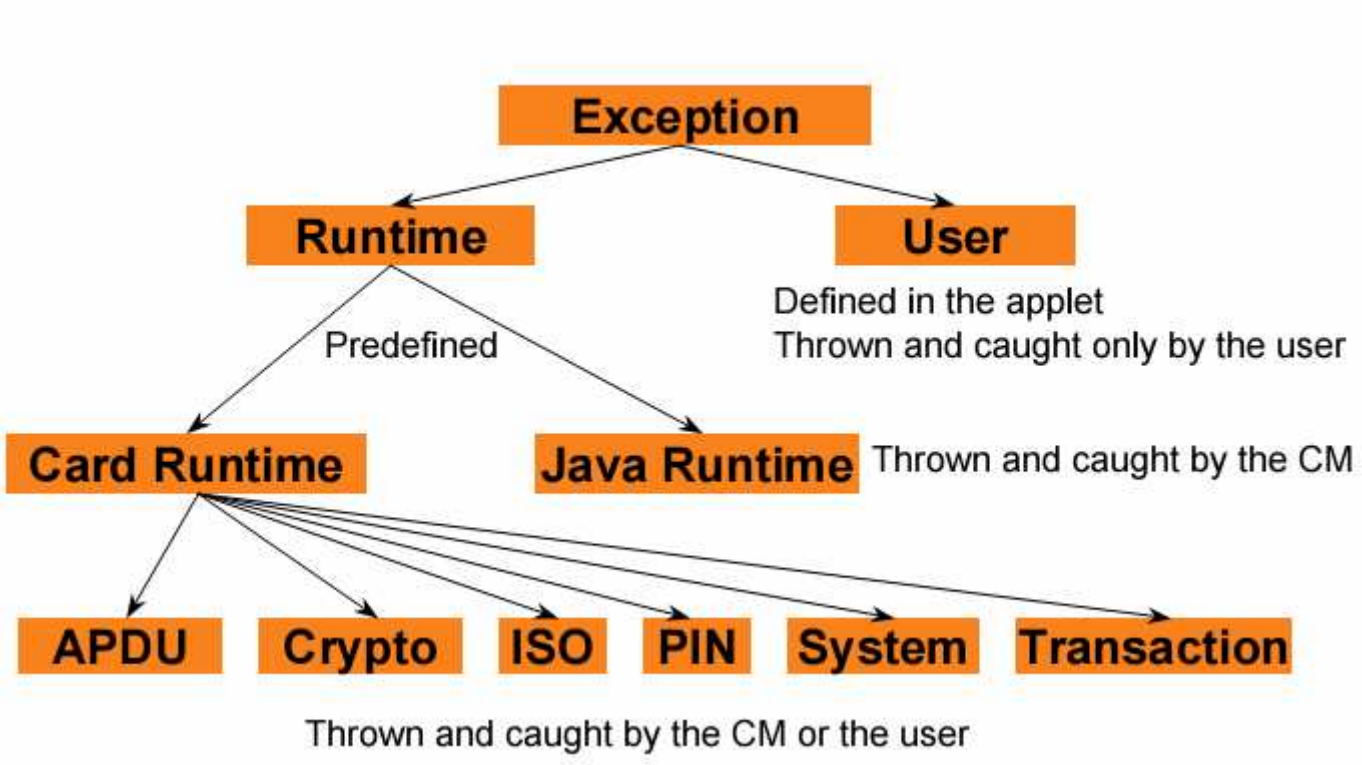


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

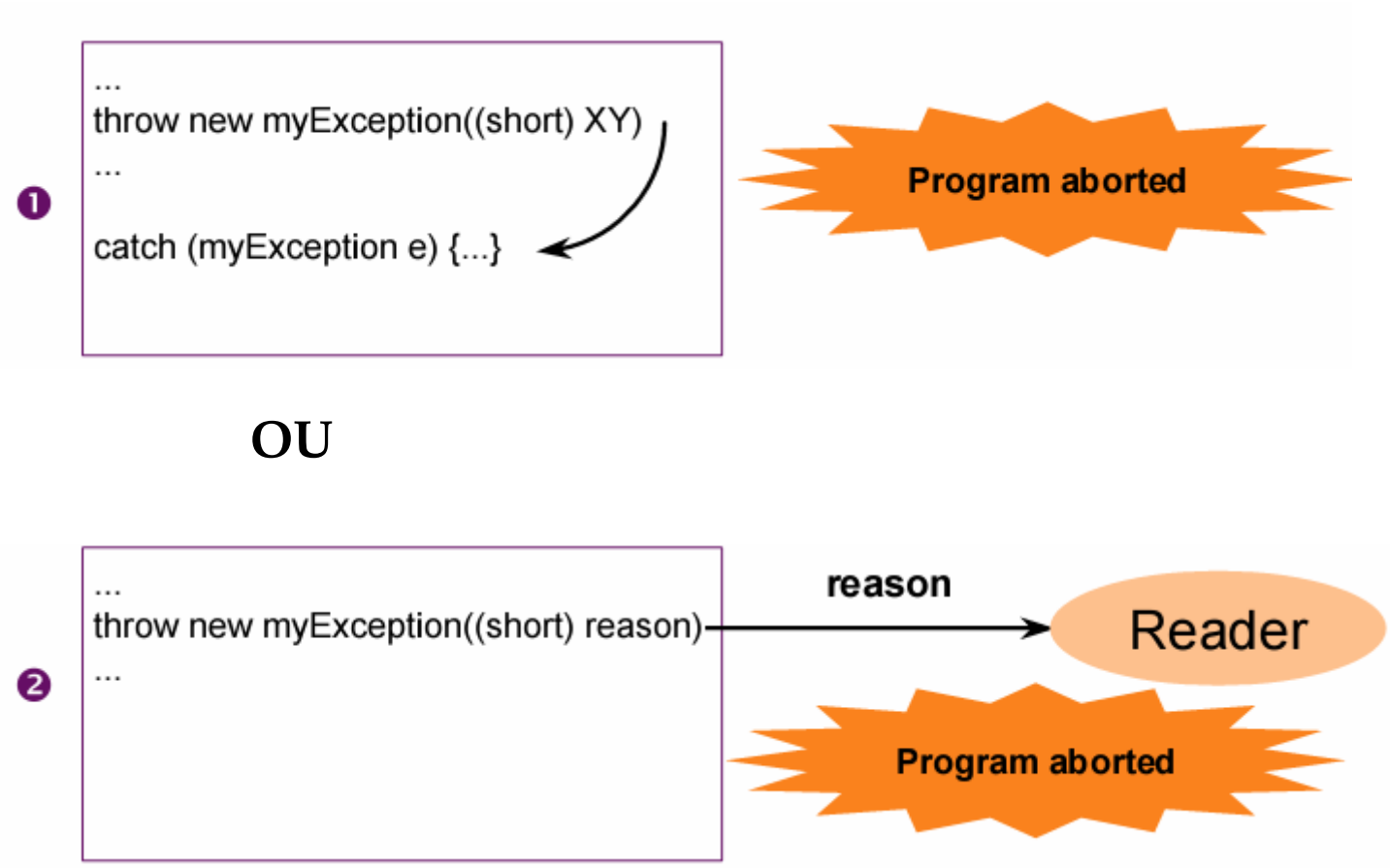
# Partage


❷ Applet B requests some shared methods from Applet A
Applet B
Applet A
❹ Applet A gives Applet B an access to the shared services
❺ Applet B can trigger Applet A shared methods
❶ Makes some methods shareable
❸ Applet A determines whether Applet B is allowed

# Card Exception



# Runtime Exception






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Exception en Java

➢Si une méthode peut lever une exception, elle doit être encapsulée par un bloc try catch.

➢ Exemple

```
try
{
    operationWhichThrowsAnException();
}
 catch (Exception e)
    {
          ….
    }
```



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Exception en Java Card

- `Exception.throwIt(value)`

- Exemple non autorisé

```
if (erreur) throw new ArithmeticException((short)0);
```



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Java Card API 2.1

- **3 paquetages de référence**
  - ✓java.lang
  - ✓javacard.framework
  - ✓javacard.security
- **Une extension**
  - ✓Javacardx.crypto (liée aux problèmes d'exportation de produits)



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Paquetage `javacard.framework`

**Class `JCSystem`**

- **Méthodes gérant l'atomicité :**
  - ✓**`beginTransaction`**(): commence la transaction
  - ✓ **`commitTransaction`**(): sauvegarde les données de la transaction dans la EEPROM
  - ✓ **`abortTransaction`**(): annule la transaction

- **Méthode de gestion d'objets temporaires**
  - ✓ **`isTransient(Object`**)
  - ✓ **`makeTransientXArray(short`**, byte) avec X= **`Boolean`**, **`Short`**, **`Object`**

- **Méthodes de gestion de partage**

- **Méthodes de gestion du système d'informations : `getVersion`()**


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Paquetage `javacard.framework`

- **Contient les spécificités de la carte**
- **La classe `Applet` :**
  - ✓Fournit un cadre à l'exécution et à l'interaction avec le JCRE de l'applet
  - ✓Les applets utilisateurs doivent étendre cette classe
- **La classe `APDU`**
  - ✓Pour la gestion des échanges avec le monde extérieur
- **La classe `PIN`**
  - ✓Assure la gestion du code secret

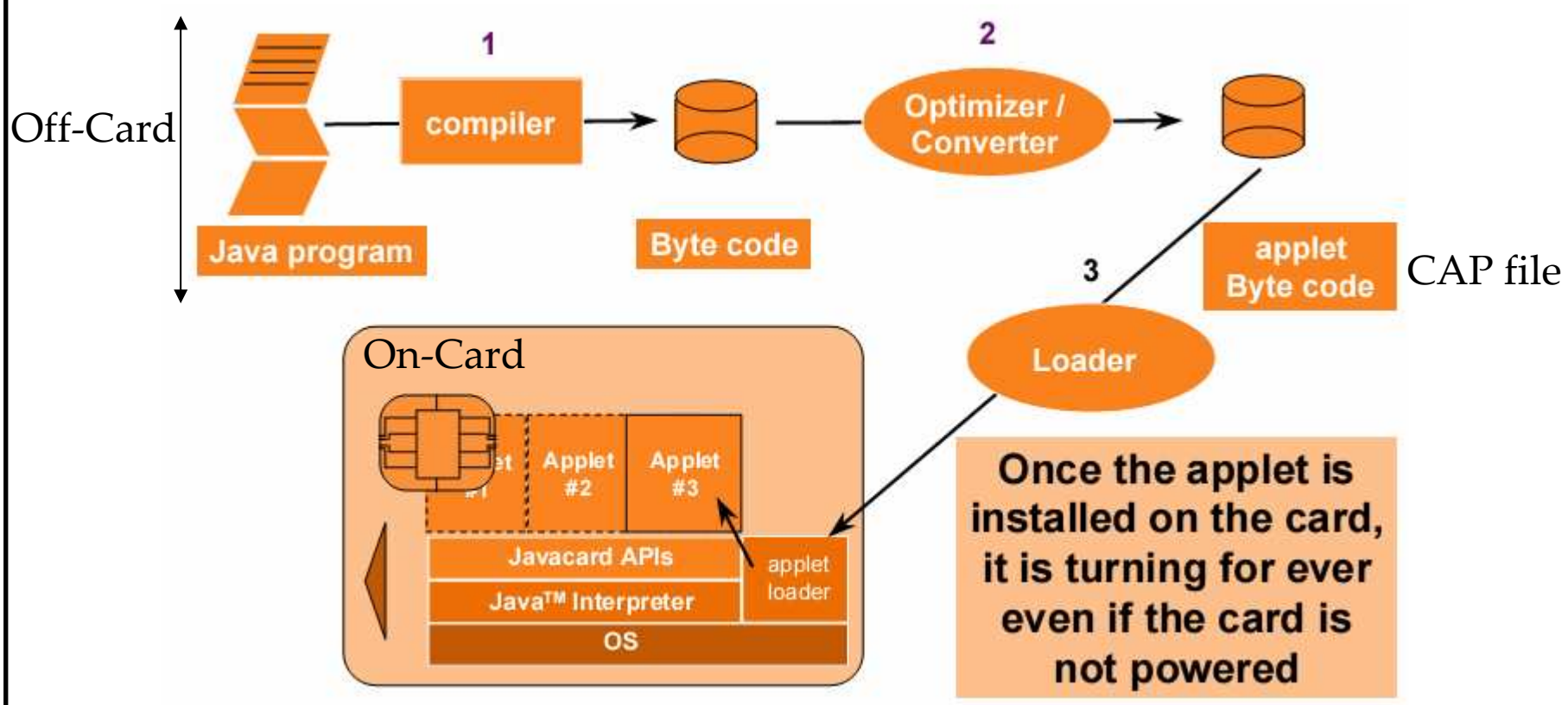

samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Paquetage `javacard.security`

- Basé sur le paquetage `java.security`
- Permet la gestion des clés et fonctions cryptographiques
- En plus des algorithmes classiques, il intègre aussi la fonction de génération de nombres aléatoires, la signature et le calcul de fonctions de compression


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Processus de développement d'applets



# CAP File

- **Le « CAP File » contient :**
  - ✓Information sur les classes
  - ✓ BC (Byte Code) exécutable
  - ✓ information nécessaire à l'édition de liens
  - ✓Information pour la vérification
- **Il est au format JAR (Java Archive), c'est ce qui est reçu par la carte**

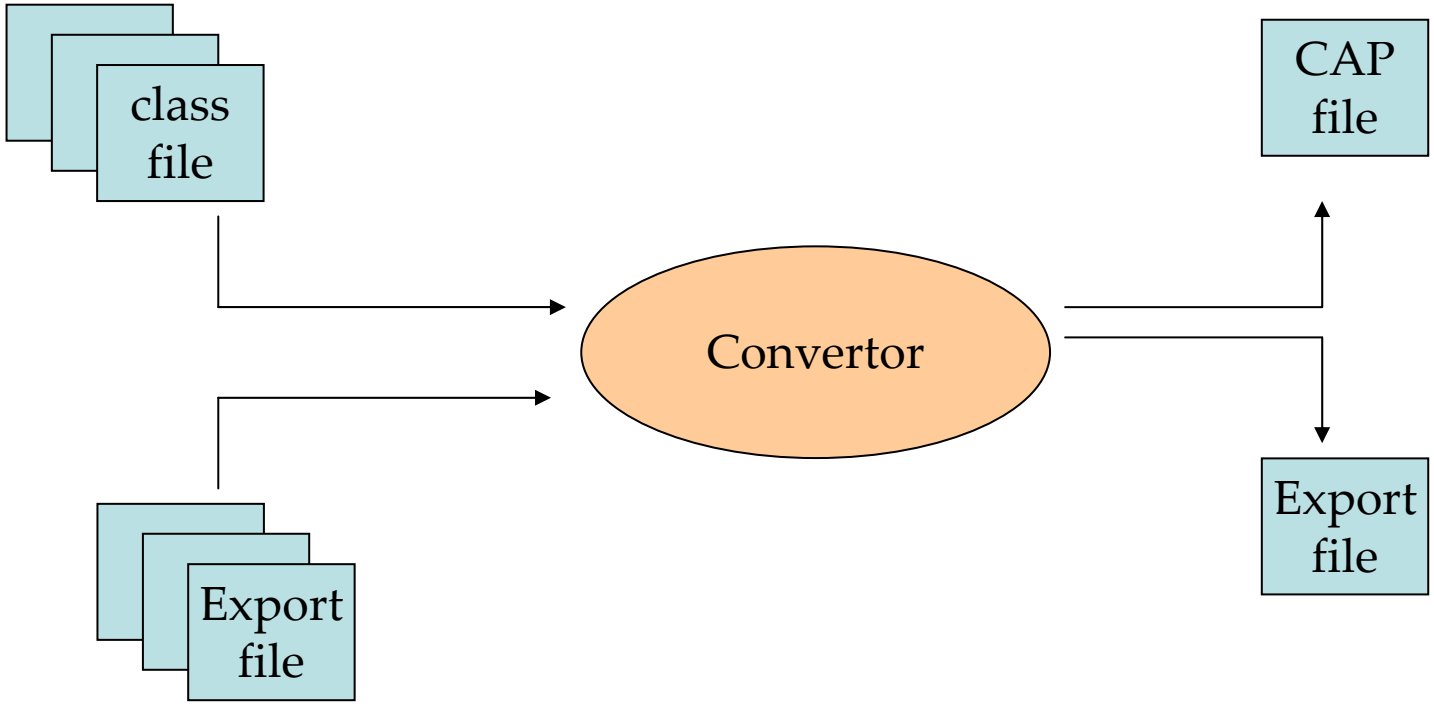

samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Export File

- **Le fichier « Export » est utilisé par le convertisseur**
- **Information utilisée pour la vérification et l'édition de liens**
- **Contient des informations publiques sur les APIs**
  - ✓Nom et champ des classes
  - ✓Signature des méthodes
  - ✓Information pour l'édition de liens entre packages
  - ✓Il ne contient pas de BC, il peut donc être publié avec une applet permettant ainsi à celle-ci d'avoir des objets utilisables (partageables)


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Le convertisseur



# Le convertisseur

➢**Prend en charge les opérations suivantes :**

✓Vérification de la conformité du format du Class File

✓Tester la conformité sur les aspects du langage Java

✓Initialisation des variables statiques

✓Résolution des références (classes, méthodes et champs) et mise sous forme compacte pour être plus efficace dans un petit système

✓Optimiser le byte code

✓Allocation et création des structures qui représentent les classes dans la JVM



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# L'interpréteur

➢**Il offre un support d'exécution au BC contenu dans le CAP file. Il permet ainsi aux applets chargées dans une carte de s'exécuter sur n'importe quelle plate-forme.**

➢ **Il réalise :**

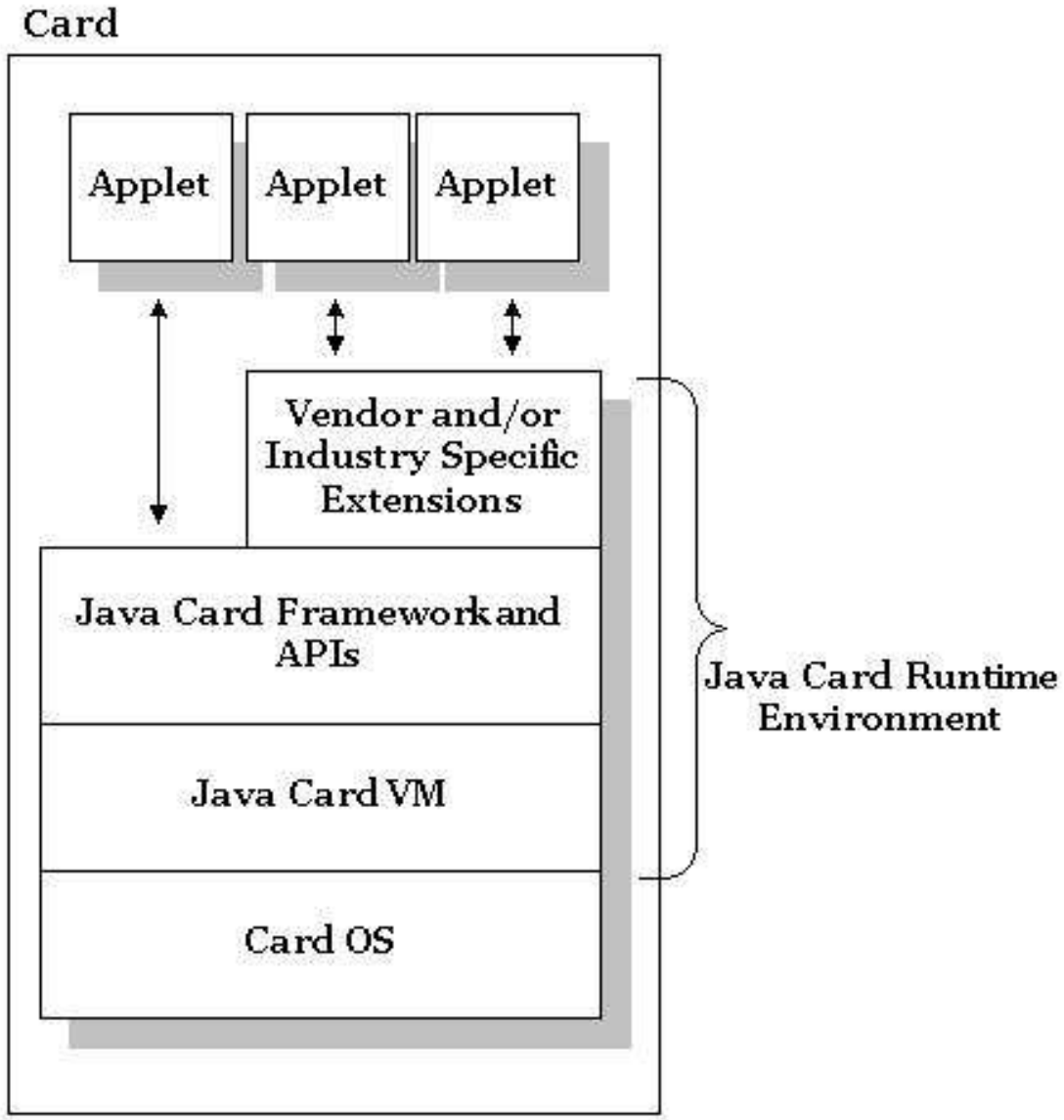
- ✓L'exécution du BC
- ✓Le contrôle de l'allocation de la mémoire
- ✓et assure la sécurité

➢ **l'installation d'applets est réalisée par un applet loader réparti sur le terminal et la carte**



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Architecture d'une Java Card






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# JCRE: cycle de vie et session carte

➢**En environnement station de travail, la JVM est un processus, elle est donc initialisée au lancement puis arrêtée à la fin du processus, les objets en RAM sont donc perdus.**

➢**Pour que les informations soient conservées d'une session à l'autre :**

✓Dans le cas de la carte, l'initialisation de la JVM est effectuée une seule fois
Au « début de la vie de la carte », les objets et les données sont conservés en mémoire non volatile (EEPROM, Flash, etc.)

✓Lors de chaque session avec la carte :

- Mise sous tension: le JCRE est « réactivé »

- La carte reçoit et traite des commandes APDU (Process_APDU)

- Mise hors tension: le JCRE est « suspendu »


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Caractéristiques du JCRE

- **Objets persistants et temporaires**

  - ✓Les objets Java Card sont par défaut persistants

  - ✓Pour des raisons d'efficacité (vitesse de lect/écrit en NVM) et sécurité (clé, résultat intermédiaire), les applets peuvent créer des objets temporaires

- **Opération atomique et transaction**

  - ✓La JCVM assure l'atomicité des écritures dans les champs d'un objet

  - ✓Le JCRE fournit une API aux applets permettant de regrouper plusieurs écritures et d'offrir la cohérence de ces écritures (Begin Transaction, Commit, Roll-Back)



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Applet firewall et mécanisme de partage

- **Chaque applet s'exécute dans un espace qui lui est propre**
  - ✓Une applet ne peut avoir d'action sur une autre applet
  - ✓La JCVM doit assurer cette caractéristique
  - ✓Il existe un mécanisme de partage, qui permet à une applet d'accéder à des services offerts par une applet ou par le JCRE.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Comment écrire une applet



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Construction d'applets Java Card

- **Une application carte**
  - ✓Code dans la carte : application serveur = Applet Java Card
  - ✓Code dans le terminal : application cliente
- **Une application construite en 3 étapes**
  - ✓ Écriture de l'application serveur (applet)
  - ✓Installation de l'applet dans la Java Card
  - ✓Écriture de l'application cliente


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Écrire une applet Java Card

- **Java Card API 2.1**
- **Étapes du développement d'une applet**
  - ✓ Spécifier les fonctions de l'applet :
    - spécifier les AIDs (Application IDentity) de l'applet et du paquetage auquel elle appartient
    - écrire le corps de l'applet
    - compiler (.class)
    - convertir (.cap)
    - charger dans la carte


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Phases de développement d'une applet

- Spécifier les fonctions de l'applet
- Assigner des AIDs à l'applet et au paquetage contenant la classe de l'applet
- Concevoir les programmes de l'applet
- Définir l'interface entre l'applet et le terminal


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Comportement d'une applet

- **Application écrite en Java Card**
- **Applet sur la carte**
  - ✓ est sélectionnée
  - ✓ reçoit des messages à partir du lecteur
  - ✓ traite ces messages
  - ✓ retourne des données au lecteur
  - ✓ est désélectionnée


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Java Card Runtime Environment

- Charge les applets sur la carte
- Sélectionne l'applet à activer
- Gère les messages reçus du lecteur (APDU)
- Gère les commandes du système de fichiers et gestionnaire de sécurité

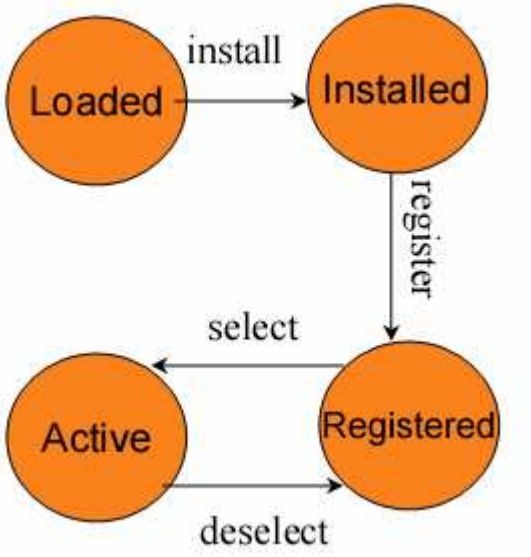

samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Cycle de vie d'une applet

Une fois l'applet chargée sur la carte, elle doit être :

- **installée**, **enregistrée** (connue du JCRE grâce à son AID)

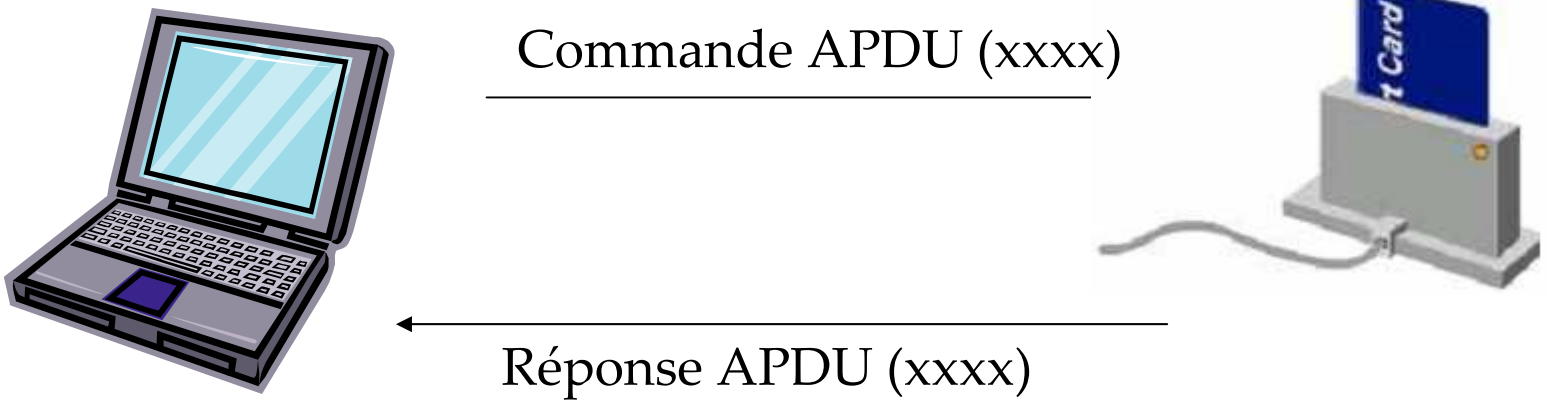
- **sélectionnée** (car plusieurs applets peuvent figurer sur la carte)



# Spécification des fonctions de l'applet

**Exemple de l'Applet Echo :**

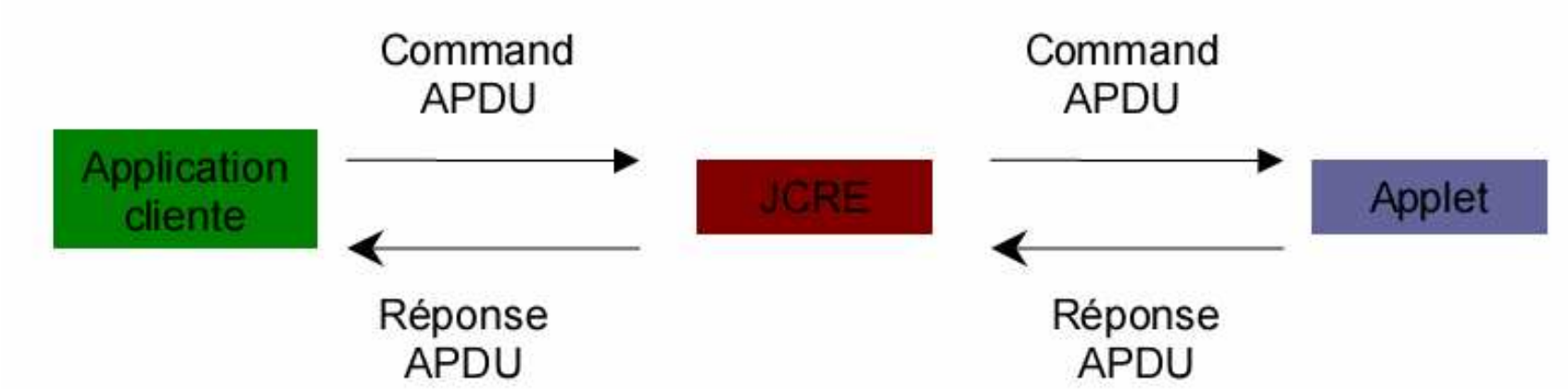
**Fonction** : Stocke la donnée qu'elle reçoit pour la retourner au terminal.






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Java Card et les Com/Rép APDU






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Spécification des AIDs

- Définir un AID pour le package et un AID pour l'Applet

| Package AID | | |
|---|---|---|
| **Champ** | **Valeur** | **Longueur** |
| RID | 0xA0, 0x00, 0x00, 0x18, 0x50 | 5 octets |
| PIX | 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x52, 0x41, 0x44, 0x50 | 10 octets (11 octets au max) |
| **Applet AID** | | |
| **Champ** | **Valeur** | **Longueur** |
| RID | 0xF2, 0x34, 0x12, 0x34, 0x56 | 5 octets |
| PIX | 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x52, 0x41, 0x44, 0x41 | 10 octets 11 octets au max) |


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Méthodes d'une Applet

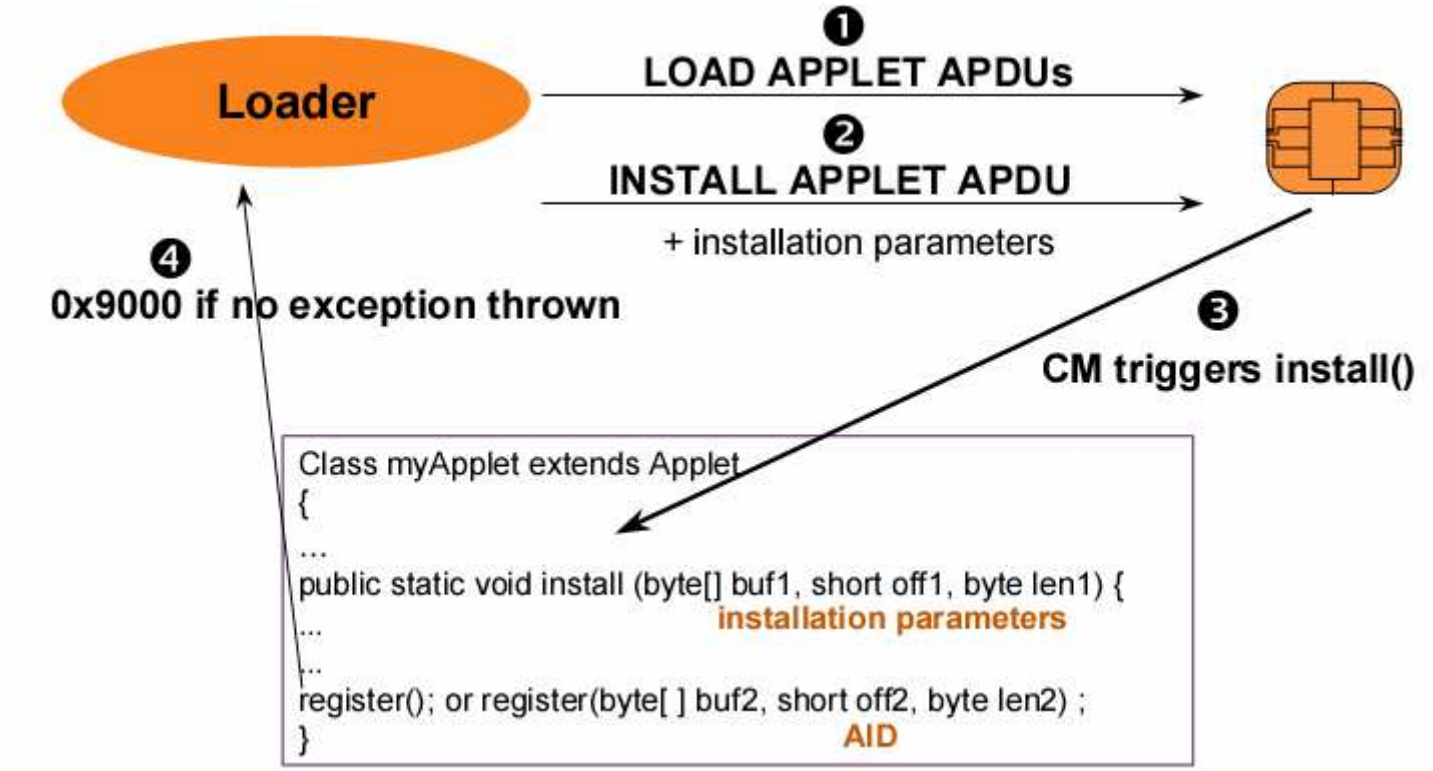
- Une applet doit toujours étendre la classe **`javacard.framework.Applet`**.

- La classe **`Applet`** définit les méthodes courantes que doit utiliser une applet pour interagir avec le JCRE.

- Ces méthodes doivent figurer dans le corps de l'applet à écrire:
  - ✓Méthodes **`select/deselect`** : pour activer/désactiver une applet
  - ✓Méthodes **`install/uninstall`**: pour installer/désinstaller l'applet sur la carte
  - ✓Méthode **`process`**: pour traiter les commandes APDU et retourner une réponse APDU
  - ✓Méthode **`register`**: pour enregistrer l'applet auprès de JCRE.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

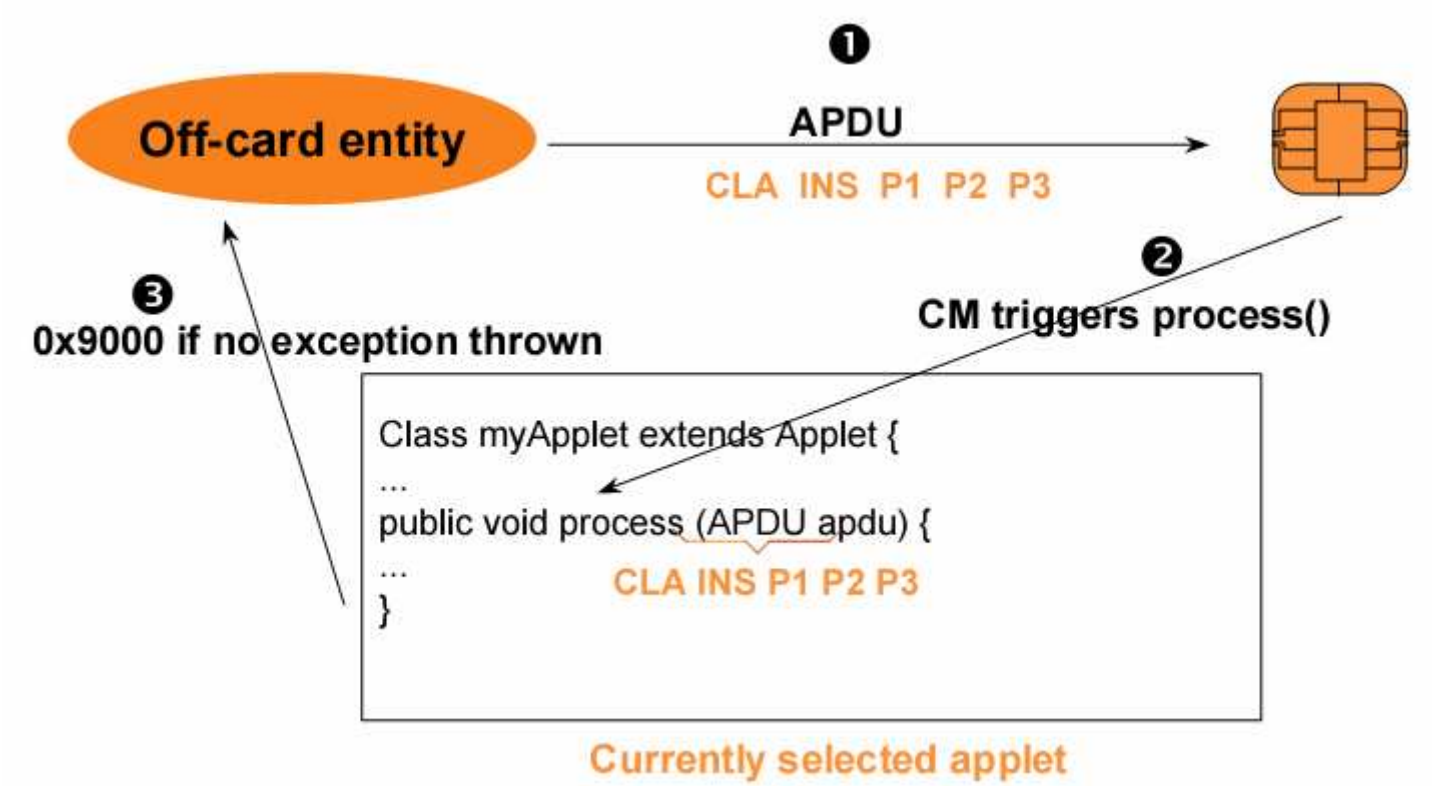
# Méthode `install`



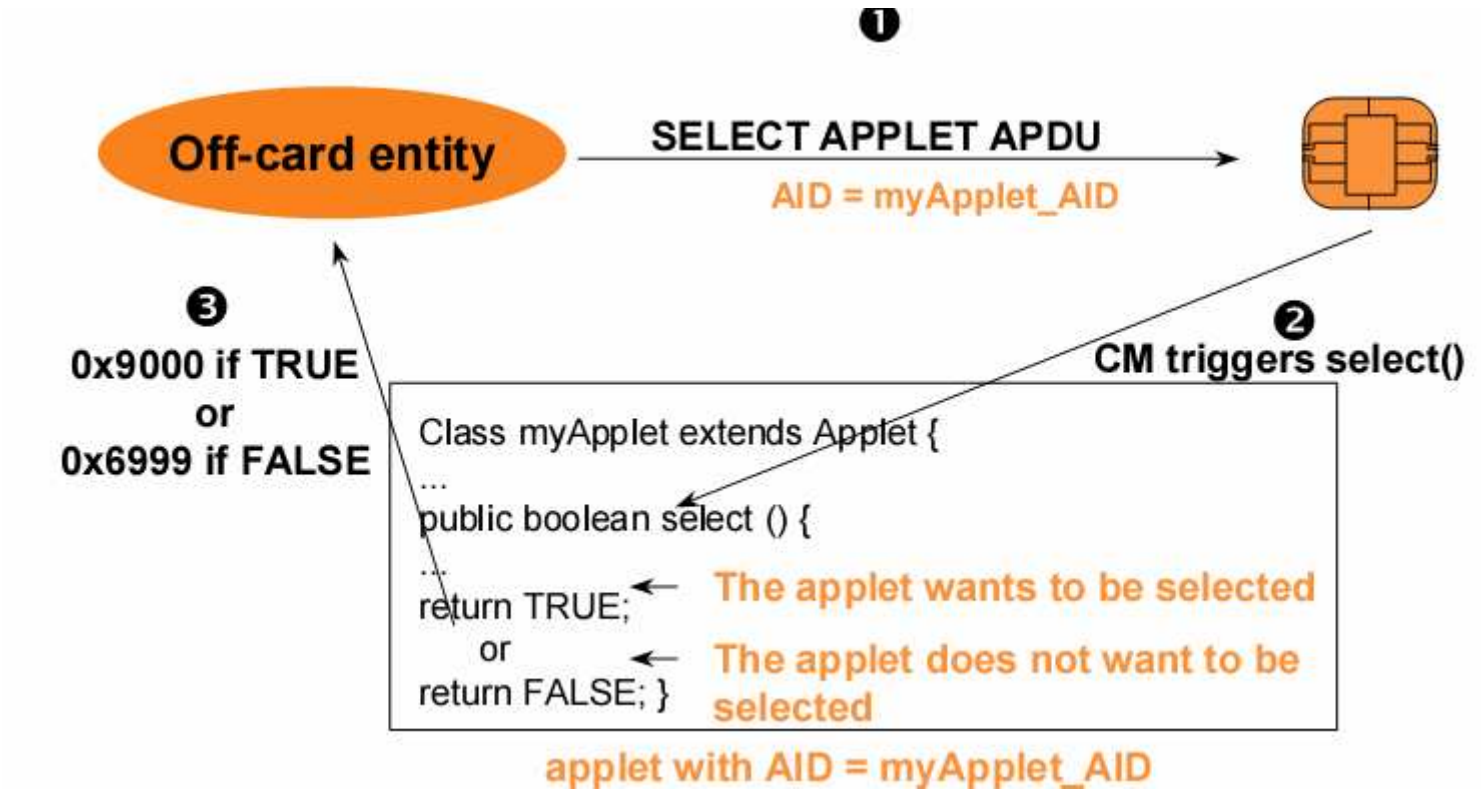



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

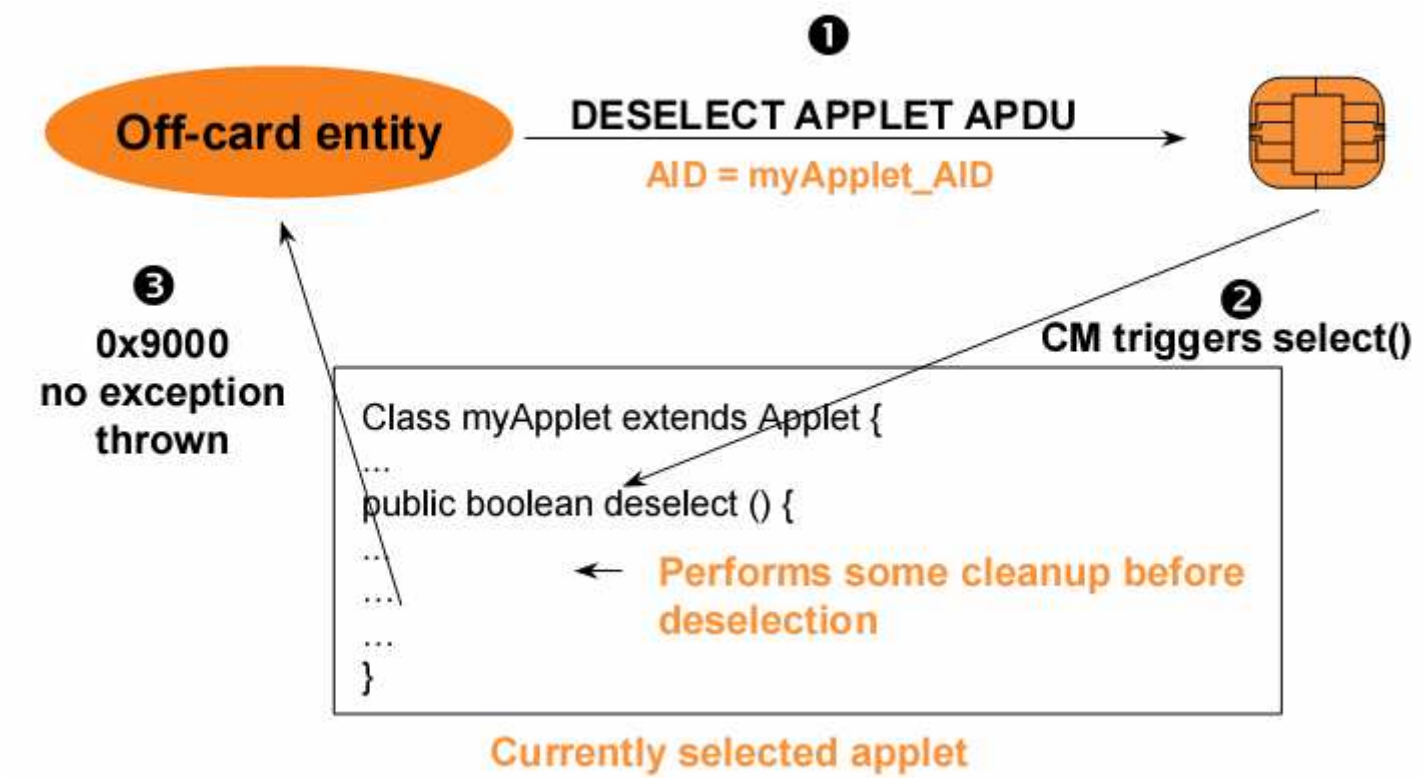
# Méthode process



# Méthode select



# Méthode deselect



# Méthodes à définir dans l'applet

| Method summary | |
|---|---|
| public void | deselect ()<br><br>Called by the JCRE to inform the currently selected applet that another (or the same) applet will be selected. |
| public Shareable | getShareableInterfaceObject (AID client AID, byte parameter)<br><br>Called by the JCRE to obtain a sharable interface object from this server applet on behalf of a request from a client applet. |
| public static void | install (byte[] bArray, short bOffset, byte bLength)<br><br>The JCRE calls this static method to create an instance of the Applet subclass. |
| public abstract void | process (APDU apdu)<br><br>Called by the JCRE to process an incoming APDU command. |
| protected final void | register ()<br><br>This method is used by the applet to register this applet instance with the JCRE and assign the default AID in the CAD file to the applet instance. |



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Méthodes à définir dans l'applet

| Method summary | |
|---|---|
| protected final void | register (byte[] bArray, short bOffset, byte bLength)<br><br>This method is used by the applet to register this applet instance with the JCRE and to assign the specified AID in the array bArray to the applet instance. |
| public boolean | select ()<br><br>Called by the JCRE to inform this applet that it has been selected. |
| protected final boolean | selectingApplet ()<br><br>This method is used by the applet process() method to distinguish the SELECT APDU command that selected this applet from all other SELECT APDU APDU commands that may relate to file or internal applet state selection. |



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Extraits de l'applet porte-monnaie

```
public void Monnaie (byte [] bArray, short bOffset, byte bLength)
       {
       balance = (short) 0x1000;
       register();
    }
    public static void install (byte [] bArray, short bOffset,
              byte bLength) {
       new Monnaie (bArray, bOffset, bLength);
    }

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

# Définir une interface entre une applet et le terminal

➢**Définir des commandes APDU:**

Une applet Java Card doit supporter un ensemble de commandes APDU:

- ✓la commande SELECT APDU : pour sélectionner une applet sur la carte
- ✓des commandes de traitement d'APDUs: commandes à traiter par la méthode process()



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Exemple de l'applet porte-monnaie électronique

Les opérations offertes par l'application :
Lire le montant du porte monnaie, débiter ou créditer son compte.

Ces opérations sont implantées sur l'applet à l'aide des méthodes suivantes : **`getBalance(), credit(), debit()`**
Ces méthodes sont appelées directement par la méthode **`process`**().

D'où: pour chaque opération définir une commande APDU qui déclenchera la méthode correspondante.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Commandes de l'applet porte-monnaie électronique

| Commande APDU CREDIT | | | | | | |
|---|---|---|---|---|---|---|
| Commande APDU | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Data field** | **Le** |
| 0xB0 | 0x30 | 0x0 | 0x0 | 1 | Valeur à créditer | NS |

| Commande APDU DEBIT | | | | | | |
|---|---|---|---|---|---|---|
| Commande APDU | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Data field** | **Le** |
| 0xB0 | 0x40 | 0x0 | 0x0 | 1 | Valeur à débiter | NS |

| Commande APDU GET BALANCE | | | | | | |
|---|---|---|---|---|---|---|
| Commande APDU | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Data field** | **Le** |
| 0xB0 | 0x50 | 0x0 | 0x0 | NS | NS | 2 |



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Traitement des commandes et réponses APDU

**1. Extraire le buffer APDU**

L'applet invoque la méthode **`getBuffer`**() pour extraire les 5 premiers octets disponibles dans le buffer : **`CLA, INS, P1, P2, et P3.`**

**2. Recevoir des données**

Si données additionnelles dans la commande, l'applet doit invoquer la méthode **`setIncomingAndReceive`**() pour diriger l'objet APDU vers la réception de données entrantes.
**`receiveBytes`**() permettra la lecture de ces données.

**3. Renvoyer des données**

**`setOutgoing`**() permet d'obtenir la longueur de la réponse (Le)
**`setOutgoingLength`**()pour informer le CAD de la longueur réelle des données à retourner.
**`sendByteLong`**() pour envoyer les données à partir du buffer.

**4. Renvoyer le mot d'état (word status)**


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Spécifications de la Java Card 2.2.2



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# La Java Card 2.2

- Canaux logiques
- suppression d'applets et de paquetages
- Mécanismes de suppression (explicite) d'objets
- Java Card Remote Method Invocation
- Support pour l'AES et des courbes elliptiques
- Support pour le sans contact


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# API de Java Card 2.2.2

- **Package java.lang**
  - Opérations arithmétiques
  - Opérations sur les vecteurs (**Array**)
  - Gestion d'exceptions,
  - etc.

- **Classes nécessaires pour faire de l'appel de méthodes à distance (RMI)**
  - ✓**Package java.rmi**
    - Remote
    - RemoteException
  - ✓**Package javacard.framework.service**
    - BasicService
    - CardRemoteObject
    - Dispatcher
    - RemoteService
    - RMIService


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# API de Java Card 2.2.2

- **Package javacard.framework**
  - AID
  - APDU
  - APDUException
  - Applet
  - ISO7816
  - ISOException
  - JCSystem
  - MultiSelectable
  - OwnerPIN
  - PIN
  - PINException
  - Util
  - etc.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# API de Java Card 2.2.2

- **Package javacard.security**
  - AESKey
  - DESKey
  - DSAKey
  - DSAPrivateKey
  - DSAPublicKey
  - ECKey
  - ECPrivateKey
  - ECPublicKey
  - HMACKey
  - KeyBuilder
  - KoreanSEEDKey
  - RSAPrivateCrtKey
  - RSAPrivateKey
  - RSAPublicKey
  - etc.

- **Package javacardx.crypto**
  - Cipher
  - KeyEncryption



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Algorithmes de cryptage proposés dans Java Card 2.2.2

- AES: Advanced Encryption Standard (FIPS-197)

- SEED Algorithm Specification : KISA - Korea Information Security Agency

Standard Names for Security and Crypto Packages
- SHA (SHA-1): Secure Hash Algorithm, as defined in Secure Hash Standard, NIST FIPS 180-1
- SHA-256,SHA-384,SHA-512: Secure Hash Algorithm,as defined in Secure Hash Standard,NIST FIPS 180-2
- MD5: The Message Digest algorithm RSA-MD5, as defined by RSA DSI in RFC 1321
- RIPEMD-160: as defined in ISO/IEC 10118-3:1998 Information technology – Security techniques - Hash-functions - Part 3: Dedicated hash-functions
- DSA: Digital Signature Algorithm, as defined in Digital Signature Standard, NIST FIPS 186
- DES: The Data Encryption Standard, as defined by NIST in FIPS 46-1 and 46-2
- RSA: The Rivest, Shamir and Adleman Asymmetric Cipher algorithm
- ECDSA: Elliptic Curve Digital Signature Algorithm
- ECDH: Elliptic Curve Diffie-Hellman algorithm
- AES: Advanced Encryption Standard (AES), as defined by NIST in FIPS 197
- HMAC: Keyed-Hashing for Message Authentication, as defined in RFC-2104


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# API de Java Card 2.2.2

- **Package javacardx.biometry**
  - BioBuilder
  - BioException
  - BioTemplate
  - OwnerBioTemplate
  - SharedBioTemplate

- **Package javacardx.framework.math**
  - BCDUtil
  - BigNumber
  - ParityBit



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# API de Java Card 2.2.2

| Core Packages | |
|---|---|
| java.io | Defines 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 | 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 technology-based applets. |
| javacard.framework.service | Provides a service framework of classes and interfaces that allow a Java Card technology-based 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 the Java Card platform. |


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

| Standard Extensions | |
|---|---|
| javacardx.apdu | Extension package that enables support for ISO7816 specification defined optional APDU related mechanisms. |
| javacardx.biometry | Extension package that contains functionality for implementing a biometric framework on the Java Card platform. |
| javacardx.crypto | Extension package that contains functionality, which may be subject to export controls, for implementing a security and cryptography framework on the Java Card platform. |
| javacardx.external | Extension package that provides mechanisms to access memory subsystems which are not directly addressable by the Java Card runtime environment(Java Card RE) on the Java Card platform. |
| javacardx.framework.math | Extension package that contains common utility functions for BCD math and parity computations. |
| javacardx.framework.tlv | Extension package that contains functionality, for managing storage for BER TLV formatted data, based on the ASN.1 BER encoding rules of ISO/IEC 8825-1:2002, as well as parsing and editing BER TLV formatted data in I/O buffers. |
| javacardx.framework.util | Extension package that contains common utility functions for manipulating arrays of primitive components - byte, short or int. |
| javacardx.framework.util.intx | Extension package that contains common utility functions for using int components. |

# Canaux logiques et Sélection d'applets

- **Session et canal logique**
  - ✓Un terminal peut ouvrir plusieurs sessions avec la carte (1 à 20 session)
  - ✓Chaque session utilise un canal logique
  - ✓Les sessions sont associées aux interfaces d'IO (il existe une interface IO pour une comm. sans contact et une interface IO pour une comm. avec contact).

- **A la mise sous tension**, le Reset est toujours envoyé sur le canal logique 0 (quelle que soit la communication avec ou sans contact).

- **Sélection d'applets**
  - ✓Une applet peut être sélectionnée sur un seul canal logique de manière à avoir plusieurs applets sélectionnées en concurrence.
  - ✓Une même applet peut être sélectionnée simultanément sur plusieurs canaux logiques.

L'octet CLA = "000 000 cc" (cc: 2 bits pour coder les numéros de canaux 0, 1, 2 et 3).
L'octet CLA = "0100 dddd" {dddd: 4 bits utilisés pour coder les numéros de canaux 4-19


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Applets Multi-sélectables

➢**Currently selected Applet** : Applet qui est en cours de traitement d'une APDU

➢**Les Applets multi-sélectables** doivent implémenter l'interface : **`javacard.framework.MultiSelectable`**

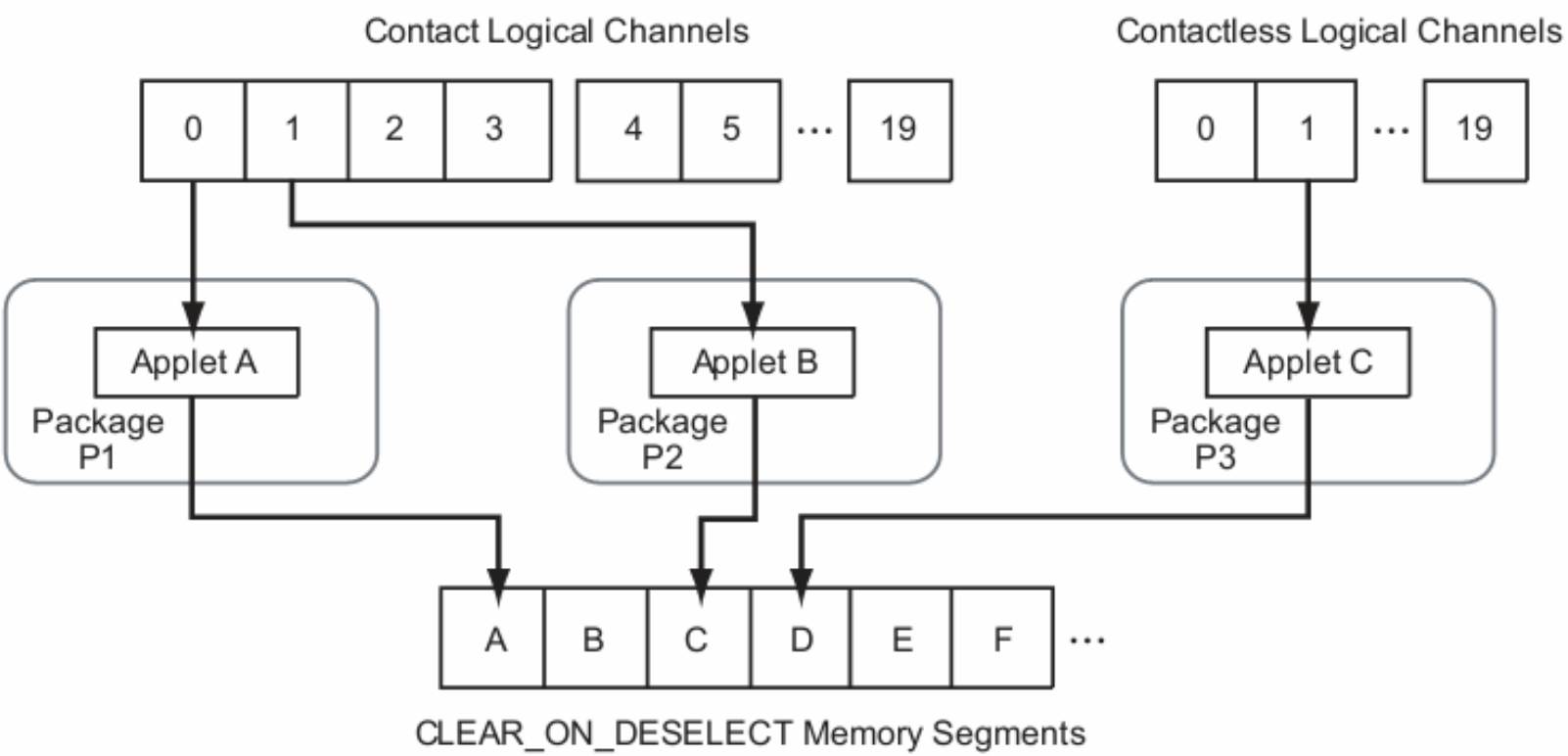
En cas de multi-sélection, les méthodes :

**`MultiSelectable.select`** () et
**`MultiSelectable.deselect`**()

sont appelées durant la sélection et la désélection respectivement.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Canaux logiques avec et/ou sans contact



# Applets par défaut

➢Normalement, une Applet ne peut être sélectionnée que si une commande SELECT FILE est reçue avec succès.

➢Mais certaines applications ont besoin d'avoir une applet sélectionnée par défaut après un Reset notamment à l'ouverture d'un  nouveau canal logique.

➢C'est l'applet par défaut.



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Comportement après un Reset (mise sous tension)

➢Après un Reset, le JCRE effectue l'initialisation et vérifie si son état interne indique qu'une applet particulière est l'applet par défaut pour le canal logique de base (canal 0).

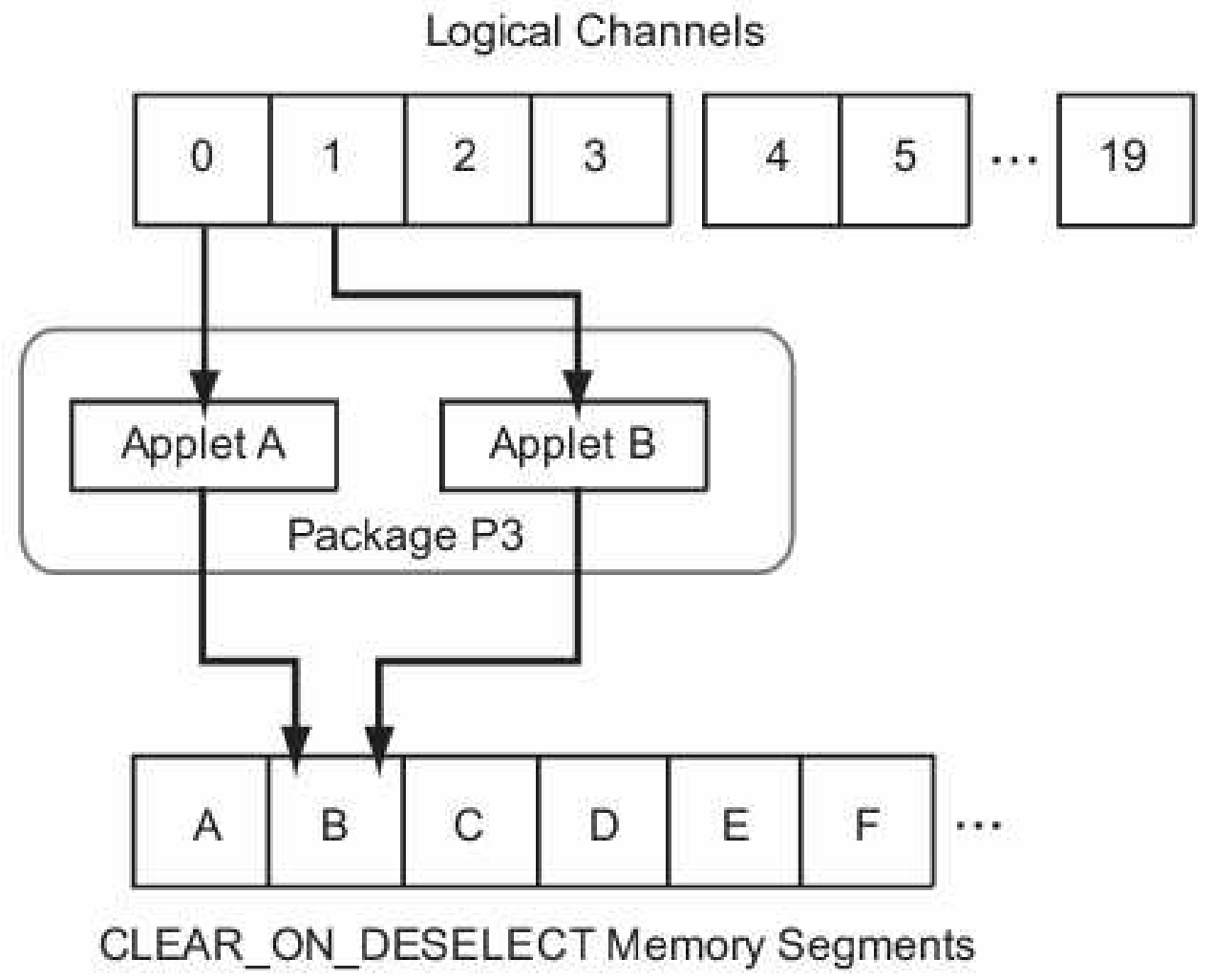
Si Oui : le JCRE sélectionne cette applet et la méthode **`select`**() de cette applet est exécutée. Si cette méthode lève une exception alors le JCRE indique qu'il n'y a pas d'applet active sur ce canal.

➢L'applet active ne se met pas en attente de réception de commandes APDUs pour activer sa méthode **`process`**() car elle n'aura pas reçu de commande APDU du type : SELECT FILE.

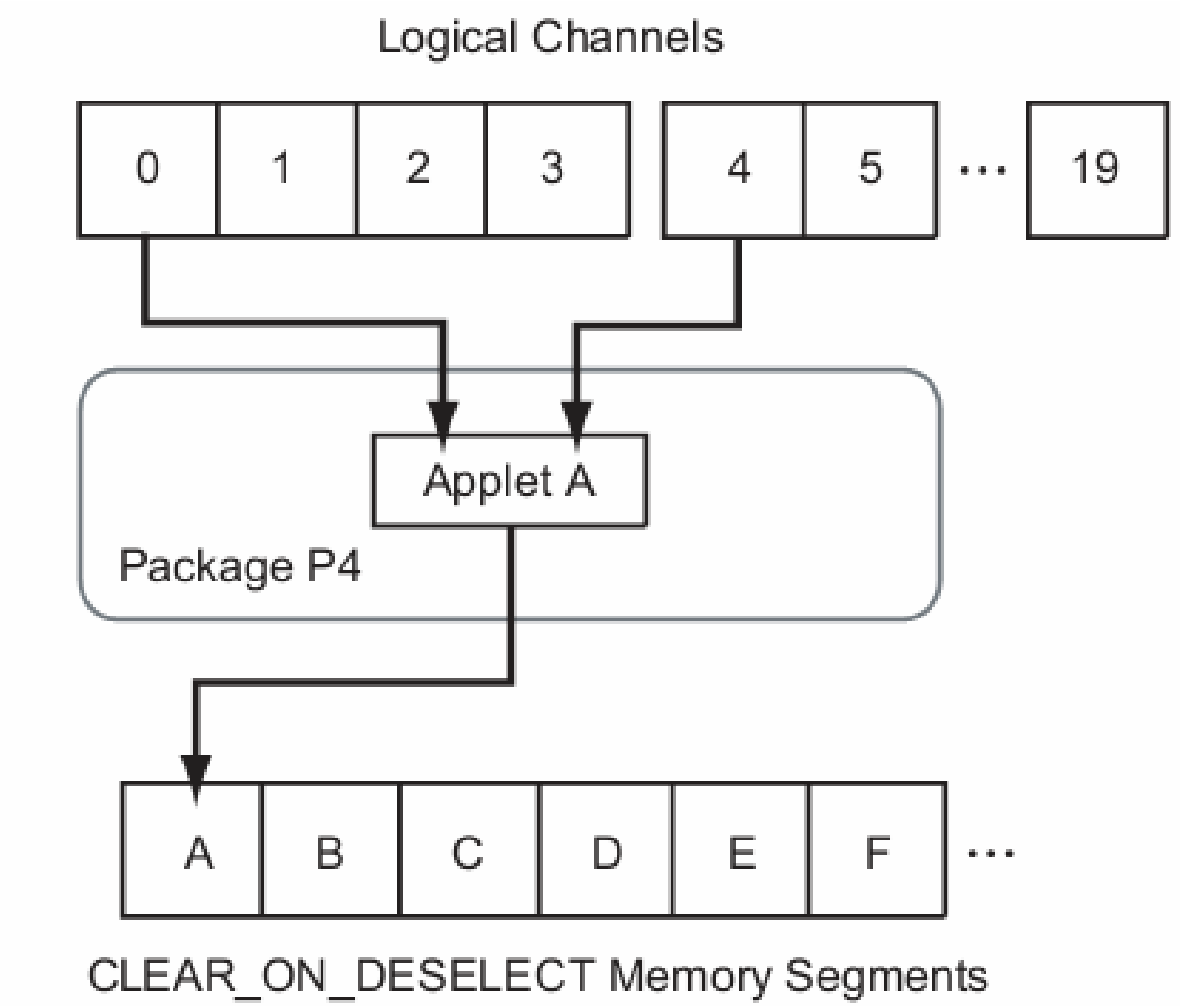
➢Le JCRE envoie un Answer to Reset (ATR) et la carte devient prête à accepter des commandes APDUs.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Plusieurs applets du même paquetage sélectionnées



# Même applet sélectionnée sur plusieurs canaux



# Deux façons de sélectionner des applets

- Avec une commande **`APDU MANAGE CHANNEL OPEN`** :
ouverture d'un canal et sélection d'une applet par défaut sur ce canal

- Avec une commande **`APDU SELECT FILE`**:

- Dans les deux cas, le **`CLA`** doit spécifier le canal ouvert ou à ouvrir

- **`INS=0x70`** pour une commande **`APDU MANAGE CHANNEL OPEN`**

- **`INS=0xA4`** pour une commande **`APDU SELECT FILE`**


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Désélection d'applet

- À la réception d'une commande **`APDU MANAGE CHANNEL CLOSE`** (**`INS=0x70`**)

- À la réception d'une commande **`SELECT FILE`** pour sélectionner une applet differente ou la même applet sur le canal spécifié dans le **`CLA`**.



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Le changement de contextes

- Des Applets Firewall partitionnent les systèmes d'objets de la Java Card en espaces d'objets protégés appelés ***contextes***.

- Tout package Java Card contient des applets qui partagent le même contexte.

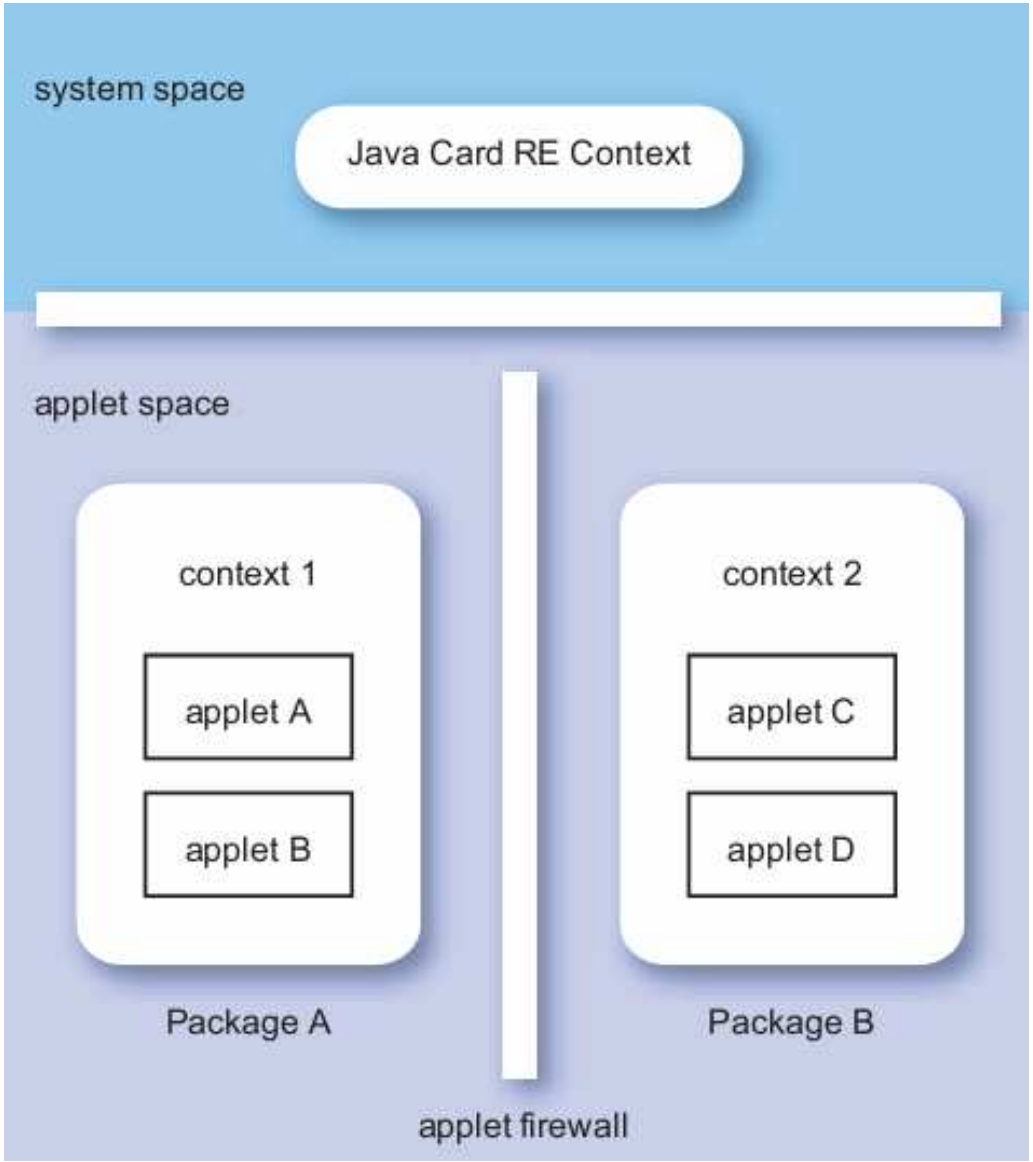
- Il n'existe pas de Firewall entre applets individuelles du même package.

  *Une applet A peut accéder aux objets d'une applet B si A et B appartiennent au même package.*



samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

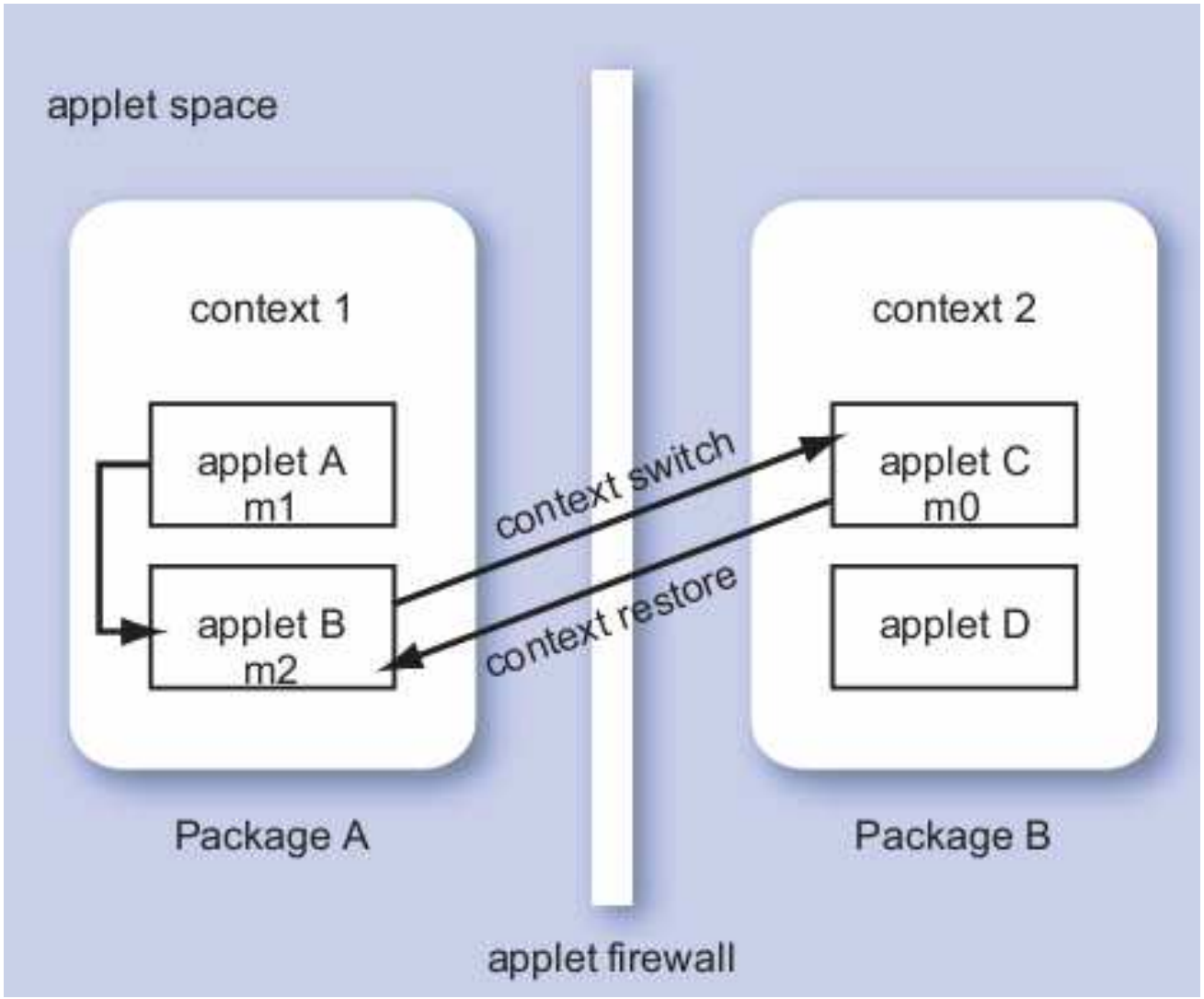
# Les contextes






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Changement de contexte






samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Conclusion

- Java Card offre aux applications de la carte un langage de programmation :
  - de haut niveau
  - comportant de bonnes propriétés (concept orienté objet).

- L'infrastructure de chargement est définie par OP (Open Platform)

- Les spécifications de la Java Card 3.0 a été publiée par SUN en mars 2008
  - édition classique (extension de la version 2.2)
  - édition orientée Web
  - intégration de la pile TCP/IP, servlets, multi-threading, etc.


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Bibliographie


1. http://java.sun.com/products/javacard/
2. http://java.sun.com/javacard/3.0/specs.jsp
2. Technology for smart cards: architecture and programmer's guide, Zhiqun Chen, Addison Wesley, sept. 2000
3. Understanding Java Card 2.0, Zhiqun Chen & Rinaldo Di Giorgio
4. http://www.javaworld.com/javaworld/jw-03-1998/jw-03-javadev.html
5. http://javacardforum.org
6. Zhiqun Chen, "**How to write a Java Card applet: A developer's guide**", http://www.javaworld.com/javaworld/jw-07-1999/jw-07-javacard.html.
7. Pierre Paradinas, Support de cours sur « **Java Card** », UV de Systèmes Enfouis et Embarqués, Valeur C, Laboratoire CEDRIC, CNAM. http://deptinfo.cnam.fr/~paradinas/cours/ValC-IntroJavaCard.pdf
8. **Global Platform, Card Specification :** http://www.globalplatform.org/specificationform2.asp?id=archived
9. **API Java Card** : http://java.sun.com/products/javacard/htmldoc
10. Eric Vétillard : http://javacard.vetilles.com/2006/09/17/hello-world-smart-card/
11. Formation dispensée par Jan Nemec de Gemalto dans le cadre du concours SIMAGINE, nov. 2007.




samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -

# Webographie

http://java.sun.com/products/javacards
http://www.gemplus.com
http://www.oberthur.com
http://www.globalplatform.org
http://www.javacardforum.org
http://www.opencard.org
http://www.linuxnet.com/
http://www.iso.org
http://www.ttfn.net/techno/smartcards/
http://eurekaweb.free.fr/ih1-carte_a_puce.htm
http://membres.lycos.fr/dbon/historique.htm
http://apte.net/info-e/pubs.htm
http://developers.sun.com/learning/javaoneonline/2008/pdf/TS-5940.pdf


samia.bouzefrane@cnam.fr - CEDRIC ( CNAM) -