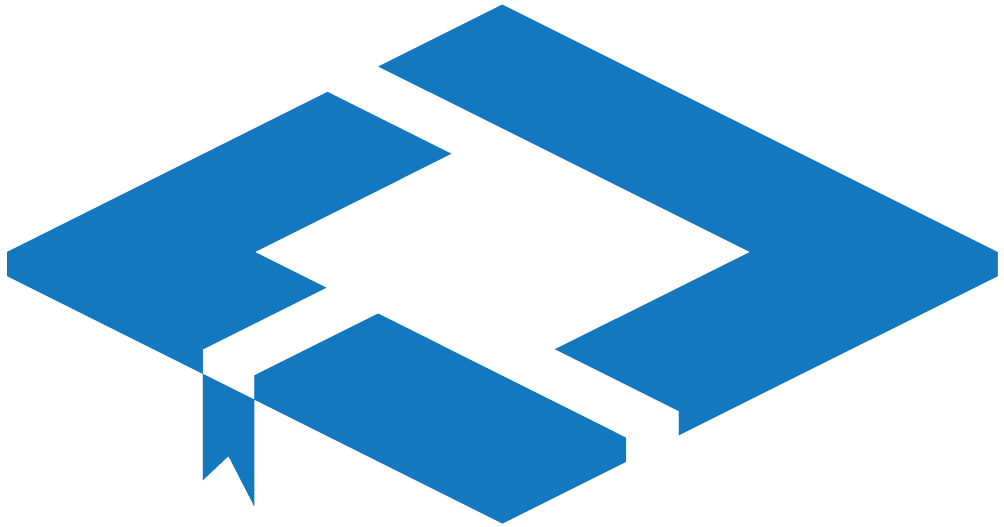
Java card



plb

Utopios® Tous droits réservés

SOMMAIRE

Utopios® Tous droits réservés

SOMMAIRE

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

L'APDU (*Application Protocol Data Unit*) est une unité de communication utilisée entre une carte Java Card et une application hôte, comme un terminal ou un lecteur de carte. Elle permet d'échanger des commandes et des réponses dans le cadre d'une communication sécurisée.

Structure d'une commande APDU

Une commande APDU est divisée en plusieurs parties selon la norme ISO/IEC 7816-4.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

1. **Header APDU (5 octets fixes)**

- **CLA** (*Class byte*) : Indique la classe de l'instruction. Par exemple, `0x00` pour des commandes ISO standard.
- **INS** (*Instruction byte*) : Spécifie l'opération à effectuer (e.g., `0xA4` pour sélectionner une application).
- **P1** et **P2** : Paramètres pour l'instruction. Leur signification dépend de la commande.
- **Lc** : Nombre d'octets de données à transmettre au lecteur.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

2. **Données (optionnel)**
   Si la commande nécessite des données d'entrée, elles sont incluses ici, immédiatement après le header.

3. **Le (optionnel)**
   Spécifie le nombre maximal d'octets attendus en réponse. Ce champ est utilisé pour demander une longueur variable de données en sortie.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

2. Structure d'une réponse APDU

- **Données (optionnel)** : Résultat ou contenu de la commande.
- **Status Word (SW1 et SW2)** : Indique l'état d'exécution de la commande. Quelques exemples :
  - `0x9000` : Succès.
  - `0x6700` : Longueur incorrecte.
  - `0x6985` : Condition d'utilisation non satisfaite.
  - `0x6A82` : Fichier ou application introuvable.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

Cycle de vie d'une communication APDU

1. **L'application hôte génère une commande APDU.**
   - Exemple : Sélection d'une application sur la carte (`CLA=0x00`, `INS=0xA4`).
2. **La carte Java Card reçoit et interprète la commande.**
3. **La carte exécute l'opération demandée.**
4. **La carte renvoie une réponse APDU contenant les données ou un code de statut.**

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

1. Commande : Sélection d'une application

```
CLA = 0x00
INS = 0xA4
P1  = 0x04
P2  = 0x00
Lc  = 0x07
Data = A0 00 00 00 03 00 00 (AID de l'application)
Le  = 0x00
```

```
Données = AID de l'application
SW1SW2 = 9000
```

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

1. Valeurs possibles pour CLA (Class Byte)

1.1 Identifier la classe de la commande.

1.2 Définir le protocole et les fonctionnalités de base.

Standard ISO/IEC 7816-4

- **0x00** : Commandes standard ISO (CLA par défaut).
- **0x10 à 0x7F** : Réservé pour des extensions futures.
- **0x80 à 0x8F** : Commandes propriétaires de l'application.
- **0x90 à 0xFE** : Commandes propriétaires inter-industrielles.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

Protocole T=1

- **0x00, 0x80, 0x84** : Utilisé fréquemment pour les commandes ISO ou spécifiques.

Bit d'encodage (selon le protocole utilisé)

- **CLA modifié pour un canal logique** :
  - Les 2 bits les plus significatifs indiquent le canal logique :
    - `0b00xx xxxx` pour le canal 0.
    - `0b01xx xxxx`, `0b10xx xxxx` : Autres canaux logiques.
  - Exemple : CLA `0x20` pour le canal 1.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

- **CLA sécurisé (protocole SM - Secure Messaging)** :
  - Le bit **SM** est activé pour indiquer un message sécurisé.

2. Valeurs possibles pour INS (Instruction Byte)

Le champ **INS** spécifie l'opération à exécuter. Les valeurs sont divisées en **commandes standard ISO** et **commandes propriétaires**.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

Commandes standard ISO/IEC 7816-4 (communes à toutes les cartes) :

INS	Commande	Description
0xA4	**SELECT**	Sélectionner un fichier ou une application
0xB0	**READ BINARY**	Lire des données dans un fichier
0xB2	**READ RECORD**	Lire un enregistrement
0xD0	**WRITE BINARY**	Écrire des données dans un fichier
0xD2	**WRITE RECORD**	Écrire un enregistrement
0x20	**VERIFY**	Vérifier un code PIN
0x24	**CHANGE REFERENCE DATA**	Modifier un code PIN
0x2C	**DISABLE VERIFICATION REQUIREMENT**	Désactiver la vérification du PIN
0x28	**ENABLE VERIFICATION REQUIREMENT**	Activer la vérification du PIN
0x88	**INTERNAL AUTHENTICATE**	Authentification interne
0x82	**EXTERNAL AUTHENTICATE**	Authentification externe
0x84	**GET CHALLENGE**	Obtenir un challenge (aléatoire)

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

Commandes propriétaires (INS propriétaires, dépend de la carte) :
Ces valeurs sont définies par les fabricants ou les applications spécifiques et ne suivent pas nécessairement les normes ISO.

- **INS propriétaires typiques** :
  - `0xE2`, `0xE4`, `0xE6`, etc. : Manipulation spécifique des fichiers.
  - `0xCA` : Obtenir des données (GET DATA).
  - `0xF2` : Commande de débogage ou de diagnostic.
  - Valeurs au-delà de **0x80** : Réservées aux applications spécifiques.

Utopios® Tous droits réservés

Le protocole APDU (Application Protocol Data Unit)

1. **Documentation du fabricant** :
   - Chaque carte (Gemalto, NXP, etc.) a sa propre liste de commandes propriétaires.
2. **Analyse des réponses APDU** :
   - Si une commande n'est pas supportée, la carte retourne généralement le code `SW1SW2 = 0x6D00` (instruction non reconnue).
3. **Exploration des fichiers et AID** :
   - En utilisant des commandes comme `SELECT` ou `GET RESPONSE`.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

1. Les bases de la communication entre la carte et le terminal

La communication entre une carte à puce et un terminal suit un **modèle maître-esclave** :

- **Le terminal** (maître) initie toutes les commandes.
- **La carte** (esclave) répond aux commandes, mais ne les initie jamais.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

2. Étapes principales de la communication

a) Initialisation

1. **Insertion ou activation de la carte** :
   - La carte est insérée dans un lecteur (contact) ou activée via NFC (sans contact).
   - Le terminal détecte la carte et établit une connexion physique.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

2. **Échange de l'ATR (Answer to Reset)** :
   - Lors de la mise sous tension, la carte envoie un **ATR** pour indiquer ses capacités au terminal.
   - L'ATR contient des informations sur :
     - Le type de protocole pris en charge (T=0, T=1).
     - Les vitesses de communication supportées.
     - D'autres paramètres spécifiques à la carte.

```
3B 9F 95 00 FF 91 81 71 FE 58
```

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

b) Établissement du protocole de communication

Les cartes à puce supportent deux principaux protocoles :

- **T=0** : Protocole basé sur des commandes, asynchrone et séquentiel.
- **T=1** : Protocole basé sur des blocs, plus rapide et structuré.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

1. **T=0 (Protocole à commande)**
   - Utilise un format simple pour envoyer une commande et attendre une réponse.
   - Chaque échange comprend :
     - Une commande APDU (envoyée par le terminal).
     - Une réponse APDU (renvoyée par la carte).

Caractéristiques :

- Simple mais moins rapide.
- Peu structuré, avec une gestion manuelle des erreurs.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

2. **T=1 (Protocole à bloc)**
   - Divise les données en blocs structurés, avec des mécanismes de contrôle d'erreurs.
   - Chaque bloc contient un en-tête, des données et un CRC (contrôle d'intégrité).

Caractéristiques :

   - Plus rapide que T=0.
   - Permet l'échange de grandes quantités de données.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

c) Échange de commandes et de réponses (APDU)

Exemple : Commande SELECT et réponse

- Commande (APDU SELECT) :

```
CLA = 0x00, INS = 0xA4, P1 = 0x04, P2 = 0x00, Lc = 0x07, Data = A0 00 00 00 03 00 01
```

- Réponse :

```
Data = [AID de l'application sélectionnée], SW1SW2 = 0x9000
```

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

d) Gestion des erreurs

Le terminal et la carte gèrent les erreurs via des **codes de statut** (SW1SW2) et des mécanismes spécifiques au protocole utilisé.

- **Exemples de codes d'erreur** :
  - `0x6A82` : Application ou fichier introuvable.
  - `0x6985` : Condition d'utilisation non satisfaite.
  - `0x6700` : Longueur incorrecte dans la commande.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

3. Mécanismes sous-jacents

a) Couche physique

- **Cartes à contact** :
  - Communication via les **broches ISO 7816**.
  - Transmission en série (un bit à la fois).
- **Cartes sans contact (NFC)** :
  - Communication via des ondes radio (ISO/IEC 14443).
  - Transmission basée sur le champ inductif généré par le terminal.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

b) Couche de transport

- Les protocoles **T=0** ou **T=1** assurent la transmission des APDU.
- Chaque commande est encapsulée dans le format du protocole choisi.

c) Couche application

- Les commandes APDU sont interprétées par l'applet ou l'OS de la carte (comme JCVM pour Java Card).
- Chaque applet traite uniquement les commandes qu'il reconnaît.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

4. Schéma global de communication

a. **Initialisation** :

- Terminal détecte la carte et négocie le protocole (T=0 ou T=1).
- La carte envoie son ATR.

b. **Sélection d'application** :

- Terminal envoie une commande SELECT avec l'AID.
- La carte répond avec les informations sur l'application.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

c. **Interaction avec l'applet** :

- Terminal envoie des commandes APDU (e.g., VERIFY PIN, READ DATA).
- La carte exécute les commandes et renvoie des réponses APDU.

d. **Déconnexion** :

- Une fois les échanges terminés, le terminal déconnecte la carte ou désactive le champ NFC.

Utopios® Tous droits réservés

Mécanismes de communication entre la carte et le terminal

5. Diagramme de communication APDU (T=0)

```
Terminal                          Carte
   |                               |
   |--- ATR ------------------->|  (Answer to Reset)
   |                               |
   |--- Commande APDU -------->|  (e.g., SELECT)
   |                               |
   |<-- Réponse APDU ---------|  (e.g., SW1SW2 = 0x9000)
   |                               |
   |--- Commande APDU -------->|  (e.g., READ DATA)
   |                               |
   |<-- Réponse APDU ---------|  (e.g., Données + SW1SW2)
   |                               |
   |--- Déconnexion ----------|
```

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

1. Prérequis à télécharger et à installer

a. Java Development Kit (JDK)

1. **Télécharger** : Oracle JDK 17 ou OpenJDK.
2. **Installer** :
   - Sous **Windows** : Téléchargez l'exécutable et suivez les instructions.
   - Sous **Linux/Mac** :

```
sudo apt install openjdk-17-jdk  # (Ubuntu/Debian)
brew install openjdk@17          # (macOS avec Homebrew)
```

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

b. Java Card Development Kit (JCDK)

1. **Télécharger** : Java Card Development Kit.
2. **Installer** :
   - Décompressez le fichier dans un répertoire, par exemple `C:\Tools\jcdk`.

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

c. Simulateur Java Card fourni par Oracle

1. **Télécharger** : Simulateur Java Card sur Oracle.
2. **Installer** :
   - Décompressez le simulateur dans un répertoire dédié, par exemple `C:\Tools\simulator`.

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

d. Outils supplémentaires

1. **GlobalPlatformPro** (facultatif, recommandé pour la gestion des clés) :
   - Télécharger ici.
   - Placez le fichier `gp.jar` dans un dossier, par exemple `C:\Tools\globalplatform`.
2. **Éditeur de code** :
   - Utilisez **IntelliJ IDEA**, **Eclipse**, ou **VSCode**.

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

2. Configuration des variables d'environnement

Ajoutez les variables suivantes à votre système :

a. **JAVA_HOME** :

- Chemin vers l'installation du JDK.
  - Windows :

```
set JAVA_HOME=C:\Program Files\Java\jdk-17
set PATH=%JAVA_HOME%\bin;%PATH%
```

  - Linux/Mac (dans `~/.bashrc` ou `~/.zshrc`) :

```
export JAVA_HOME=/usr/lib/jvm/java-17
export PATH=$JAVA_HOME/bin:$PATH
```

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

b. **JC_HOME** :

- Répertoire du Java Card Development Kit :

```
JC_HOME=C:\Tools\jcdk
PATH=%JC_HOME%\bin;%PATH%
```

b. **JC_HOME_SIMULATOR** :

- Répertoire du simulateur Java Card :

```
JC_HOME_SIMULATOR=C:\Tools\simulator
```

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

3. Configurer le simulateur

a. Injecter les clés SCP

L'outil `Configurator.jar`, fourni avec le simulateur, permet de configurer les clés SCP (Secure Channel Protocol).

1. **Commande pour injecter les clés** :

```
java -jar C:\Tools\simulator\tools\configurator.jar -binary C:\Tools\simulator\runtime\bin\jcsw.exe -SCP-keyset 01 \
    51671DB062BEA84D91CA2963C2A8E951 2AC8CD7000E83DAF85BB3A4721DEF0F1 5EE5DF028861B5649B58D7166A50F270 -force
```

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

2. **Explications des paramètres** :
   - `-binary` : Chemin vers l'exécutable du simulateur.
   - `-SCP-keyset 01` : Définit un Key Version Number (**KVN**) et les clés SCP.
   - `-force` : Permet d'écraser les configurations précédentes.

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

b. Démarrer le simulateur

Lancez le simulateur après avoir injecté les clés :

```
java -jar C:\Tools\simulator\bin\jcwde.jar
```

Logs attendus :

- Le simulateur doit indiquer qu'il est prêt à écouter sur le port `9025`.

Utopios® Tous droits réservés

Mise en place de l'environnement de développement

4. Tester le simulateur avec un client

a. Configurer un fichier `client.config.properties`

1. Exemple de fichier :

```
A000000151000000_scp03enc_10=51671DB062BEA84D91CA2963C2A8E951
A000000151000000_scp03mac_10=2AC8CD7000E83DAF85BB3A4721DEF0F1
A000000151000000_scp03dek_10=5EE5DF028861B5649B58D7166A50F270
```

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

Qu’est-ce que Java Card ?

Java Card est une plateforme technologique qui permet d’exécuter des applications Java sur des cartes à puce. Elle est conçue pour répondre aux contraintes des environnements embarqués avec des ressources limitées en termes de mémoire et de puissance de calcul.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

1. **Association de Java et des cartes à puce :**
   - Une Java Card combine la puissance du langage Java avec la sécurité et les fonctionnalités d'une carte à puce.
   - Le langage Java, orienté objet, permet un développement simplifié et structuré des applications pour cartes à puce.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

2. **Indépendance des composants :**
   - Séparation entre l'application, le système d'exploitation et le matériel.
   - Contrairement aux cartes traditionnelles où l'application est liée au système propriétaire, Java Card permet à n'importe quel développeur Java de créer des applets.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

3. **Cycle de développement réduit :**
   - Avec Java Card, le cycle de développement d'une application est raccourci, passant de plusieurs mois (environ 5 mois pour une carte traditionnelle) à environ 2 mois.
4. **Multi-applications :**
   - Les Java Cards peuvent gérer plusieurs applications sur la même carte, avec des données et du code associés.
   - Cela permet de maximiser l'utilisation des ressources de la carte.

Utopios® Tous droits réservés

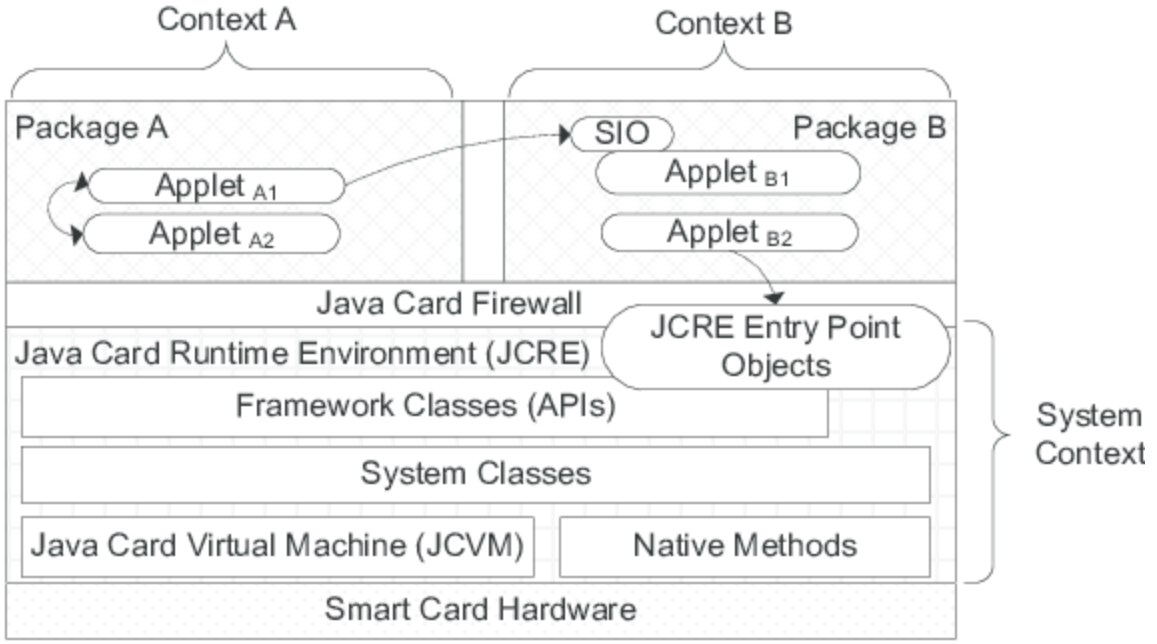
Présentation de la Technologie Java Card

5. **Interopérabilité :**
   - Les applets développés sur une plateforme Java Card sont compatibles avec plusieurs autres plateformes Java Card, ce qui garantit une grande flexibilité.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

Éléments de base d'une Java Card :



Utopios® Tous droits réservés

Présentation de la Technologie Java Card

Éléments de base d’une Java Card :

1. **Java Card Runtime Environment (JCRE) :**
    - Assure l'exécution des applets.
    - Gère les interactions avec le terminal via des commandes APDU (Application Protocol Data Unit).

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

Éléments de base d'une Java Card :

2. **Java Card Virtual Machine (JCVM) :**
   - Version compacte de la JVM classique, adaptée aux contraintes de mémoire des cartes.
   - Permet d'exécuter du bytecode Java.
3. **Java Card API :**
   - Fournit des bibliothèques spécifiques pour la gestion des applets, de la sécurité (cryptographie), et des interactions avec l'extérieur.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

Exemples d'utilisation :

1. **Cartes SIM :**
   - Une grande majorité des cartes SIM utilisées dans les téléphones mobiles fonctionnent avec Java Card.
2. **Cartes bancaires :**
   - Support pour les transactions sécurisées et les fonctions multi-applications.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

Exemples d'utilisation :

3. **Authentification et sécurité :**
   - Utilisées pour des systèmes nécessitant un haut niveau de sécurité, comme les badges d'accès ou les passeports électroniques.

Utopios® Tous droits réservés

Présentation de la Technologie Java Card

- **Sécurité :** Langage sécurisé, gestion de la mémoire et des accès via le "firewall" d'applet.
- **Flexibilité :** Ajout ou mise à jour d'applications sans avoir à remplacer la carte physique.
- **Compatibilité :** Les applications peuvent être utilisées sur plusieurs cartes compatibles Java Card.
- **Facilité de développement :** Utilisation de Java, un langage de haut niveau, standardisé et bien documenté.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Le **Forum Java Card** et **Oracle** jouent un rôle clé dans le développement, la normalisation et la promotion de la technologie Java Card. Voici une explication détaillée de leur rôle et de leur contribution.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

1. Le Forum Java Card (Java Card Forum - JCF)

- C’est une association internationale réunissant les principaux acteurs de l’écosystème Java Card, notamment :
  - Les fabricants de cartes à puce (silicium).
  - Les intégrateurs (encarteurs).
  - Les entreprises clientes utilisant des Java Cards.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Objectifs du Forum Java Card :

1. **Promouvoir la technologie Java Card :**
   - Mettre en avant les avantages de la technologie Java Card dans les différents secteurs (télécommunications, finance, etc.).
   - Développer des cas d'utilisation adaptés aux besoins des entreprises.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Objectifs du Forum Java Card :

2. **Définir les orientations technologiques :**
   - Les membres du forum discutent et élaborent des spécifications techniques pour améliorer Java Card.
   - Proposer ces spécifications à Sun Microsystems (aujourd'hui Oracle), qui les intègre dans les standards officiels.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Objectifs du Forum Java Card :

3. **Favoriser l'interopérabilité :**
   - S'assurer que les cartes Java Card développées par différents fabricants respectent les mêmes normes et peuvent fonctionner avec les mêmes applets.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Impact du Forum Java Card :

- **Innovation :** Le JCF a contribué à l'introduction de fonctionnalités clés dans les versions récentes de Java Card, comme le support des algorithmes de cryptographie avancés (AES, courbes elliptiques) ou la gestion du sans contact.
- **Normalisation :** Grâce à son rôle consultatif, le JCF aide à définir les standards industriels pour Java Card, garantissant une adoption large et homogène.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

2. Le Rôle d'Oracle dans Java Card

Contexte historique :

- **Sun Microsystems** est l'inventeur de la technologie Java et le propriétaire initial des spécifications Java Card.
- En 2010, Oracle a racheté Sun Microsystems, devenant ainsi le propriétaire des technologies Java, y compris Java Card.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Contributions d'Oracle :

1. **Normalisation et Propriété des Spécifications :**
   - Oracle publie les spécifications officielles de Java Card, basées sur les contributions du JCF.
   - Les spécifications sont accessibles publiquement sur le site officiel Java Card.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Contributions d'Oracle :

2. **Mise à disposition des outils de développement :**
   - Oracle fournit une **implémentation de référence** pour Java Card, permettant aux développeurs de tester leurs applets sur une plateforme de simulation.
   - Une **suite de tests de compatibilité** est disponible pour s'assurer que les cartes et applets respectent les normes Java Card.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Contributions d'Oracle :

3. **Licence Java Card Technology :**
   - Oracle propose une licence pour les fabricants et développeurs souhaitant produire des cartes ou applets Java Card :
     - **Compatibilité garantie** : Les produits licenciés peuvent porter le logo "Java Powered".
     - **Support technique** : Oracle offre un support dédié aux licenciés pour faciliter le développement et l'intégration de la technologie.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

Contributions d'Oracle :

4. **Évolution de Java Card :**
   - Oracle a introduit Java Card 3.0, divisée en deux éditions :
     - **Édition Classique** : Extension des versions précédentes.
     - **Édition Connectée** : Intégration de nouvelles fonctionnalités comme la prise en charge de la pile TCP/IP, des servlets, et du multi-threading.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

3. Collaboration entre le Forum Java Card et Oracle

- **Oracle** agit comme l'autorité centrale qui finalise et publie les spécifications Java Card, basées sur les propositions du **Forum Java Card**.
- Cette collaboration garantit que les avancées technologiques sont intégrées dans un cadre standardisé, accessible à l'ensemble des acteurs de l'industrie.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

4. Avantages pour l'écosystème Java Card

1. **Pour les développeurs :**
   - Des outils robustes et des spécifications claires.
   - Une interopérabilité assurée entre les différentes plateformes Java Card.
2. **Pour les fabricants :**
   - Des normes définies pour créer des cartes compatibles avec une large gamme d'applications.
   - Accès à des outils de test et de certification.

Utopios® Tous droits réservés

Le Forum Java Card et Oracle

5. Ressources associées :

- **Forum Java Card** : http://javacardforum.org
- **Oracle Java Card Technology** : http://java.sun.com/products/javacard/

Utopios® Tous droits réservés

Java Card : Un Sous-ensemble de Java

Java Card est une version simplifiée et adaptée de Java conçue spécifiquement pour fonctionner dans des environnements avec des contraintes de ressources, tels que les cartes à puce. Bien qu'elle repose sur les principes fondamentaux du langage Java, elle diffère de manière significative du Java Development Kit (JDK) classique en raison des limitations imposées par l'environnement matériel.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

1. Gestion des Threads

- **Java Card :**
  - **Pas de support pour les threads.**
  - L'exécution sur une Java Card est strictement séquentielle et gérée par le système d'exploitation de la carte.
  - Cette limitation réduit la complexité de la gestion des tâches concurrentes et évite les problèmes liés à l'allocation de ressources pour le multithreading.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

1. Gestion des Threads

- **JDK Classique :**
  - Supporte le multithreading avec l'API `java.lang.Thread` et les mécanismes associés (synchronisation, moniteurs).
  - Permet une gestion fine des processus parallèles et multitâches.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

2. Gestion de la Mémoire

- **Java Card :**
  - Utilise une mémoire très limitée (typiquement : 1 Ko de RAM, 24-28 Ko de ROM et 8-16 Ko d'EEPROM).
  - Pas de **garbage collector** (collecteur de déchets) jusqu'à la version 2.2. Cela signifie que la mémoire allouée pour les objets doit être gérée manuellement ou est persistante jusqu'à la réinitialisation de la carte.
  - Les objets peuvent être persistants (stockés dans l'EEPROM) ou temporaires (stockés dans la RAM).

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

2. Gestion de la Mémoire

- **JDK Classique :**
  - Inclut un garbage collector qui gère automatiquement la mémoire.
  - Prend en charge une mémoire bien plus vaste et permet des allocations dynamiques complexes.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

3. Support des Types de Données

- **Java Card :**
  - Limité à des types de données simples pour réduire la consommation de mémoire et la complexité du système :
    - Types pris en charge : `boolean`, `byte`, `short`, et un support optionnel pour `int`.
    - Pas de support pour : `long`, `float`, `double`, ni les types complexes comme les génériques ou les collections standard (ex. `ArrayList`).

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

3. Support des Types de Données

- **JDK Classique :**
  - Supporte tous les types primitifs (`int`, `long`, `float`, etc.) et offre une riche bibliothèque de types complexes comme les collections (`List`, `Map`, `Set`).

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

4. Modèle d'Exécution

- **Java Card :**
  - Utilise une machine virtuelle optimisée appelée **Java Card Virtual Machine (JCVM)**.
  - Le bytecode Java Card est plus compact et adapté aux environnements à mémoire restreinte.
  - Certaines fonctionnalités avancées de la JVM classique, comme le chargement dynamique de classes, ne sont pas disponibles.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

4. Modèle d'Exécution

- **JDK Classique :**
  - Utilise la JVM complète qui permet l'exécution de bytecode Java standard.
  - Supporte le chargement dynamique de classes et la réflexion.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

5. Modèle de Sécurité

- **Java Card :**
  - Intègre un **firewall d'applet** pour isoler les différentes applets qui coexistent sur une même carte.
  - Offre une API de cryptographie avancée pour des opérations sécurisées (AES, RSA, SHA, etc.).

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

5. Modèle de Sécurité

- **JDK Classique :**
  - Utilise une gestion de sécurité via des politiques (fichiers `policy`) et des gestionnaires de permissions (`SecurityManager`).

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

6. Fonctionnalités du Langage

- **Java Card :**
  - Pas de **chargement dynamique** de classes.
  - Pas de **clonage** (`Object.clone()`).
  - Pas de **sérialisation** (`java.io.Serializable`).
  - Pas de tableaux multidimensionnels (seulement des tableaux à une dimension).

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

6. Fonctionnalités du Langage

- **JDK Classique :**
  - Prend en charge toutes les fonctionnalités ci-dessus, ce qui le rend beaucoup plus flexible pour les applications complexes.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

7. Cycle de Vie et Stockage des Objets

- **Java Card :**
  - Les objets Java Card sont par défaut persistants. Ils restent en mémoire même après une mise hors tension.
  - Les objets temporaires sont stockés en RAM et sont réinitialisés à certains événements (ex. réinitialisation de la carte).
- **JDK Classique :**
  - Les objets sont volatils et dépendent de la durée de vie du programme ou du garbage collector.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

1. **Contraintes Matérielles :**
   - Les cartes à puce disposent de ressources limitées en termes de mémoire, puissance de calcul et capacité de stockage.
   - Les fonctionnalités non essentielles pour les applications embarquées sont retirées.
2. **Simplicité et Sécurité :**
   - En limitant les fonctionnalités (pas de threads, pas de chargement dynamique), Java Card réduit la surface d'attaque potentielle et simplifie le développement.

Utopios® Tous droits réservés

Différences entre Java Card et le JDK classique

3. **Focus sur la Cryptographie :**
   - Java Card est souvent utilisée pour des tâches critiques en matière de sécurité (authentification, chiffrement, signature numérique). Elle intègre des API spécifiques pour ces besoins.

Utopios® Tous droits réservés

Création d’une Applet Java Card

Une applet Java Card est une application légère conçue pour s'exécuter dans l'environnement restreint d'une carte à puce. Elle doit respecter une structure bien définie et utiliser les API spécifiques à Java Card. Voici une explication détaillée des concepts, avec des exemples pratiques pour chaque étape.

Utopios® Tous droits réservés

Structure d'une Applet Java Card

A. Structure de Base

Une applet Java Card est essentiellement une classe qui étend la classe abstraite `javacard.framework.Applet`. Elle doit implémenter certaines méthodes clés pour interagir avec le Java Card Runtime Environment (JCRE).

Utopios® Tous droits réservés

Structure d’une Applet Java Card

B. Méthodes Principales

1. **Méthode `install()`** :
   - Méthode statique appelée par le JCRE lors de l'installation de l'applet dans la carte.
   - Permet de créer une instance de l’applet.
   - Exemple :

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

Utopios® Tous droits réservés

Structure d'une Applet Java Card

2. **Méthode `process()`** :
   - Gère les commandes reçues sous forme d'APDU (Application Protocol Data Unit).
   - Appelée chaque fois qu'une commande APDU est envoyée à l'applet.

```
public void process(APDU apdu) {
    byte[] buffer = apdu.getBuffer();
    short bytesRead = apdu.setIncomingAndReceive();
    apdu.setOutgoing();
    apdu.setOutgoingLength(bytesRead);
    apdu.sendBytes((short) 0, bytesRead);
```

Utopios® Tous droits réservés

Structure d'une Applet Java Card

3. **Méthodes `select()` et `deselect()`** (optionnelles) :
   - `select()` : appelée lorsque l'applet est sélectionnée.
   - `deselect()` : appelée lorsque l'applet est désélectionnée.

Utopios® Tous droits réservés

Structure d'une Applet Java Card

C. Exemple Complet : Applet Echo

Cet exemple illustre une applet qui renvoie les données reçues.

```
import javacard.framework.*;

public class EchoApplet extends Applet {
    public static void install(byte[] bArray, short bOffset, byte bLength) {
        new EchoApplet();
    }

    public void process(APDU apdu) {
        if (selectingApplet()) {
            return; // Ignore la commande SELECT
        }

        byte[] buffer = apdu.getBuffer();
        short bytesRead = apdu.setIncomingAndReceive();

        apdu.setOutgoing();
        apdu.setOutgoingLength(bytesRead);
        apdu.sendBytes((short) 0, bytesRead);
    }
}
```

Utopios® Tous droits réservés

Extension avec des Packages Spécifiques

A. Gestion des Transactions

Java Card offre une API pour gérer des transactions atomiques, garantissant qu'un ensemble d'opérations est soit entièrement exécuté, soit annulé en cas d'échec.

Utopios® Tous droits réservés

Extension avec des Packages Spécifiques

1. **Exemple d'utilisation des Transactions** :
   - Méthodes : `JCSystem.beginTransaction()`, `JCSystem.commitTransaction()`, `JCSystem.abortTransaction()`.

```
public void debit(APDU apdu) {
    JCSystem.beginTransaction();
    try {
        // Logique métier, par exemple, débiter un compte
        balance -= amount;
        JCSystem.commitTransaction();
    } catch (Exception e) {
        JCSystem.abortTransaction();
    }
}
```

Utopios® Tous droits réservés

Extension avec des Packages Spécifiques

2. **Applications typiques :**
   - Gestion de portefeuille électronique.
   - Sauvegarde cohérente des données sensibles.

Utopios® Tous droits réservés

Extension avec des Packages Spécifiques

B. Cryptographie avec `javacard.security`

1. **Présentation** :
   - Le package `javacard.security` fournit des classes pour gérer les clés et les algorithmes cryptographiques.
   - Prise en charge des algorithmes comme AES, DES, RSA, SHA, HMAC.

Utopios® Tous droits réservés

Extension avec des Packages Spécifiques

2. **Génération de clés RSA :**

Exemple d'une applet qui génère une paire de clés RSA.

```
import javacard.security.*;
import javacard.framework.*;

public class CryptoApplet extends Applet {
    private KeyPair rsaKeyPair;

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

    public CryptoApplet() {
        rsaKeyPair = new KeyPair(KeyPair.ALG_RSA, KeyBuilder.LENGTH_RSA_1024);
        rsaKeyPair.genKeyPair(); // Génération des clés
    }
    public void process(APDU apdu) {
        byte[] buffer = apdu.getBuffer();
        if (buffer[ISO7816.OFFSET_INS] == (byte) 0x20) { // Exemple d'instruction
            PublicKey pubKey = rsaKeyPair.getPublic();
            // Renvoyer la clé publique au terminal
            apdu.setOutgoing();
            apdu.setOutgoingLength((short) pubKey.getSize());
            pubKey.getW(buffer, (short) 0);
            apdu.sendBytes((short) 0, (short) pubKey.getSize());
        }
    }
}
```

Utopios® Tous droits réservés

Extension avec des Packages Spécifiques

3. **Utilisation de HMAC pour vérifier l'intégrité :**
   - Création et utilisation de clés HMAC :

```
import javacard.security.*;

HMACKey hmacKey = (HMACKey) KeyBuilder.buildKey(
    KeyBuilder.TYPE_HMAC, KeyBuilder.LENGTH_HMAC_SHA_256, false
);
hmacKey.setKey(hmacKeyData, (short) 0);

Signature hmacSignature = Signature.getInstance(Signature.ALG_HMAC_SHA_256, false);
hmacSignature.init(hmacKey, Signature.MODE_SIGN);
```

Utopios® Tous droits réservés

Étapes de Développement et Test

1. **Compilation et Conversion :**
   - Compiler le fichier `.java` pour obtenir un fichier `.class`.
   - Utiliser l'outil **Converter** pour transformer les fichiers `.class` en un fichier CAP (Java Card Converted Applet File).
2. **Chargement dans la Carte :**
   - Utiliser un outil comme **GlobalPlatformPro** ou **GPShell** pour charger le fichier CAP dans la carte.

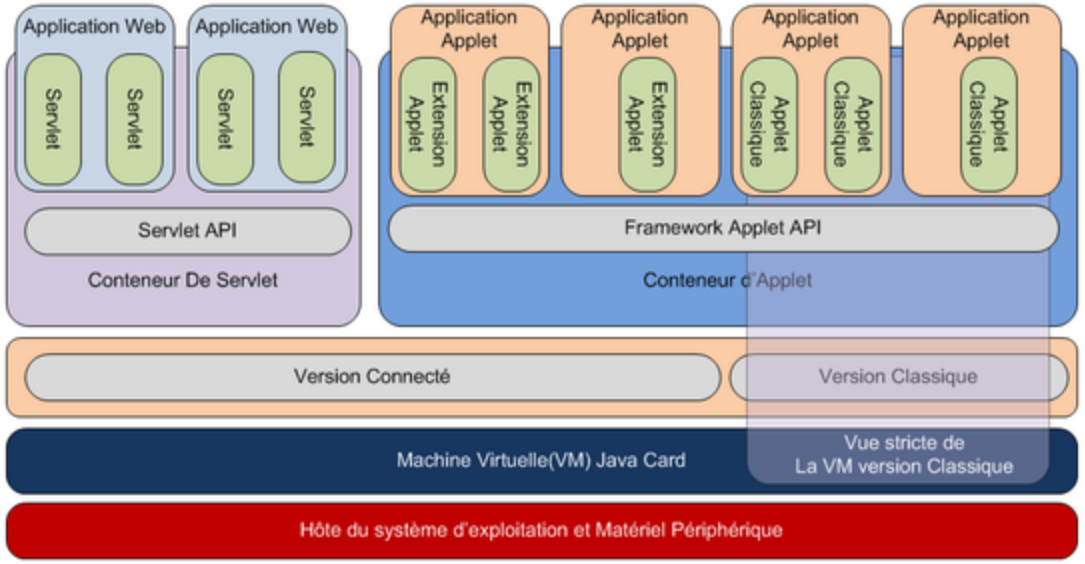
Utopios® Tous droits réservés

Étapes de Développement et Test

3. **Test avec des Commandes APDU :**
   - Envoyer des commandes APDU au moyen d'un simulateur (ex. JCOP) ou d'un terminal connecté.

Utopios® Tous droits réservés

Java card 3.x


Application Web
Servlet
Servlet
Application Web
Servlet
Servlet
Servlet API
Conteneur De Servlet
Application Applet
Extension Applet
Extension Applet
Application Applet
Extension Applet
Application Applet
Applet Classique
Applet Classique
Application Applet
Applet Classique
Framework Applet API
Conteneur d'Applet
Version Connecté
Version Classique
Machine Virtuelle(VM) Java Card
Vue stricte de
La VM version Classique
Hôte du système d'exploitation et Matériel Périphérique

Technologie Java Card 3.0 (Édition Connectée)

La version **Java Card 3.0** (édition connectée) marque une avancée majeure en introduisant des fonctionnalités modernes qui élargissent les possibilités des cartes Java Card. Elle vise à prendre en charge des cas d'utilisation plus complexes grâce à des fonctionnalités avancées telles que les servlets, la gestion des transactions, le multi-threading et le partage d'objets.

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

A. Présentation

- Les **servlets** sont des composants Java qui permettent d'exécuter des applications côté serveur directement dans l'environnement Java Card.
- Cela signifie que la carte peut répondre à des requêtes HTTP/S, transformant une Java Card en un véritable serveur embarqué.

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

B. Fonctionnalités Clés

1. **Prise en charge de HTTP/HTTPS** :
   - La carte peut traiter les requêtes HTTP, y compris les GET, POST, et PUT.
   - La pile TCP/IP est intégrée dans l'édition connectée.
2. **Support des servlets Java classiques** :
   - Les servlets Java sont écrits de manière similaire aux servlets dans des environnements classiques, comme Java EE.

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

C. Exemple d'implémentation

Voici un exemple simple de servlet pour répondre à une requête HTTP :

```
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        resp.setContentType("text/plain");
        resp.getWriter().write("Hello from Java Card Servlet!");
    }
}
```

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

D. Cas d'utilisation

- **Authentification distante** : La carte peut agir comme un serveur d'authentification.
- **Mise à jour de données** : Les données sensibles sur la carte peuvent être accessibles et modifiables via des interfaces web sécurisées.

Utopios® Tous droits réservés

2. Gestion des Transactions : Atomicité et Rollback

A. Atomicité

- Une transaction est **atomique** si toutes les opérations qu'elle contient sont exécutées avec succès ou sont toutes annulées en cas d'échec.
- Java Card 3.0 assure une **gestion transactionnelle** robuste pour garantir l'intégrité des données sensibles.

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

B. Rollback

- Si une transaction ne se termine pas normalement (ex. interruption de courant ou retrait de la carte), toutes les modifications effectuées pendant la transaction sont annulées.
- Cela garantit qu'aucune donnée ne reste dans un état incohérent.

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

C. API pour les Transactions

1. **Début de la transaction :**

```
JCSystem.beginTransaction();
```

2. **Validation des modifications :**

```
JCSystem.commitTransaction();
```

3. **Annulation des modifications :**

```
JCSystem.abortTransaction();
```

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

D. Exemple

Une applet permettant de débiter un montant du solde en garantissant l'atomicité :

```
public void debit(short amount) {
    JCSystem.beginTransaction();
    try {
        if (balance >= amount) {
            balance -= amount;
            JCSystem.commitTransaction();
        } else {
            JCSystem.abortTransaction();
            ISOException.throwIt(ISO7816.SW_CONDITIONS_NOT_SATISFIED);
        }
```

Utopios® Tous droits réservés

Servlets : Exécution d'applications côté serveur

E. Cas d'utilisation

- **Paiements sécurisés** : Garantir qu'un montant débité est soit totalement retiré, soit pas du tout.
- **Mises à jour de données sensibles** : Éviter les erreurs lors de la mise à jour des données critiques, comme un numéro de compte ou des clés cryptographiques.

Utopios® Tous droits réservés

Multi-threading : Exécution simultanée des opérations

A. Présentation

- Java Card 3.0 (édition connectée) introduit le **multi-threading**, permettant à plusieurs threads d'exécuter des opérations simultanément.
- Cela améliore les performances et la réactivité des applications complexes.

Utopios® Tous droits réservés

Multi-threading : Exécution simultanée des opérations

B. Fonctionnement

1. **Gestion des threads :**
    - Les threads sont gérés par la machine virtuelle Java Card.
    - Chaque thread peut exécuter une tâche distincte, comme traiter des requêtes HTTP ou gérer des transactions simultanément.

Utopios® Tous droits réservés

Multi-threading : Exécution simultanée des opérations

2. **Synchronisation des objets :**
   - Les mécanismes de synchronisation Java classiques (`synchronized`, moniteurs) sont disponibles pour éviter les conflits d'accès.

Utopios® Tous droits réservés

Multi-threading : Exécution simultanée des opérations

C. Exemple

Un exemple où deux threads partagent l'accès à une ressource :

```
public class MultiThreadedApplet {
    private short sharedResource;

    public synchronized void incrementResource() {
        sharedResource++;
    }

    public synchronized short getResource() {
        return sharedResource;
    }
}
```

Utopios® Tous droits réservés

Multi-threading : Exécution simultanée des opérations

D. Cas d'utilisation

- **Traitement des transactions en parallèle** : Plusieurs utilisateurs peuvent interagir avec une carte en même temps.
- **Serveur embarqué** : Traiter plusieurs requêtes HTTP simultanément.

Utopios® Tous droits réservés

Partage d’objets : Partage d’objets entre applets

A. Présentation

- Java Card 3.0 permet à plusieurs applets de partager des objets.
- Ce mécanisme est essentiel pour les cartes contenant plusieurs applets qui doivent collaborer, comme une applet de paiement et une applet de fidélité.

Utopios® Tous droits réservés

Partage d’objets : Partage d’objets entre applets

B. Mécanisme de Partage

1. **Interface Shareable** :
   - Les objets partagés doivent implémenter l’interface `Shareable`.
   - Exemple :

```
public interface ShareableObject extends Shareable {
    void doSomething();
}
```

Utopios® Tous droits réservés

Partage d'objets : Partage d'objets entre applets

2. **Accès aux Objets Partagés :**
   - Une applet cliente peut obtenir une référence à un objet partagé via la méthode `JCSystem.getAppletShareableInterfaceObject`.
   - Exemple :

```
Shareable sharedObject = JCSystem.getAppletShareableInterfaceObject(serverAID, parameter);
```

Utopios® Tous droits réservés

Partage d'objets : Partage d'objets entre applets

C. Exemple

Un objet partagé entre une applet de paiement et une applet de fidélité :

1. **Applet Serveur :**

```
public class PaymentApplet extends Applet {
    private Shareable sharedResource;

    public Shareable getShareableInterfaceObject(AID clientAID, byte parameter) {
        return sharedResource;
    }
}
```

Utopios® Tous droits réservés

Partage d’objets : Partage d’objets entre applets

2. Applet Cliente :

```
Shareable resource = JCSystem.getAppletShareableInterfaceObject(serverAID, (byte) 0);
resource.doSomething();
```

Partage d’objets : Partage d’objets entre applets

D. Cas d’utilisation

- **Cartes multi-applications** :
  - Une applet de fidélité partage des points avec une applet de paiement.
  - Une applet de gestion d’accès partage des données avec une applet de sécurité.

Utopios® Tous droits réservés

Résumé java card 3.x

Fonctionnalité	Description	Cas d'utilisation
Servlets	Exécution de requêtes HTTP/S sur la carte.	Authentification distante, gestion des mises à jour.
Transactions	Atomicité des opérations avec support du rollback.	Paiements sécurisés, mises à jour critiques.
Multi-threading	Gestion de threads pour des opérations simultanées.	Serveurs embarqués, interactions parallèles.
Partage d'objets	Collaboration entre applets via des objets partagés.	Cartes multi-applications (paiement, fidélité, etc.).

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

Le développement sécurisé dans Java Card implique de maximiser l'efficacité et de minimiser les risques en respectant des contraintes matérielles strictes (mémoire limitée, faible puissance de calcul).

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

1. Programmation Efficace et Économique

A. Réduction de la Consommation Mémoire

1. **Utilisation de la RAM pour les Données Temporaires** :
   - Java Card distingue deux types de mémoire :
     - **EEPROM** : Non volatile, mais avec un nombre limité d'écritures (coût élevé en performance).
     - **RAM** : Volatile, mais rapide.
   - Stockez les données temporaires en RAM pour améliorer les performances.

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

2. **Minimisation des Écritures en EEPROM** :
   - Limitez les modifications en mémoire non volatile pour réduire l'usure.
   - Préférez des mises à jour en lot si possible.

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

- Exemple inefficace :

```
for (byte i = 0; i < 10; i++) {
    eepromData[i] = i; // Écriture répétée
}
```

- Exemple optimisé :

```
Util.arrayCopyNonAtomic(tempData, (short) 0, eepromData, (short) 0, (short) 10);
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

3. **Réutilisation des Objets** :
   - Réutilisez les instances au lieu d'en créer de nouvelles pour éviter la fragmentation mémoire.
   - Exemple :

```
private byte[] reusableBuffer = new byte[256]; // Réutilisable
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

B. Optimisation des Structures de Données

1. **Utilisation de Types de Données Simples** :
   - Java Card prend en charge des types simples (`byte`, `short`), adaptés aux contraintes matérielles.
   - Évitez l'utilisation de `int` sauf si absolument nécessaire.

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

2. **Réduction de la Taille des Buffers** :
   - N'allouez que la mémoire nécessaire pour les données.
   - Exemple :

```
byte[] smallBuffer = new byte[64]; // Suffisant pour des APDU standards
```

3. **Compression des Données** :
   - Stockez des données de manière compacte.
   - Exemple : Utiliser un tableau de bits pour représenter des drapeaux au lieu d'un tableau de bytes.

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

2. Gestion des Exceptions et Optimisation du Code

A. Gestion des Exceptions

1. **Évitez les Exceptions dans les Situations Prévisibles** :
   - Les exceptions consomment des ressources et ralentissent les performances.
   - Préférez des validations explicites pour gérer les cas normaux.

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

- Exemple inefficace :

```
try {
    array[index] = value;
} catch (ArrayIndexOutOfBoundsException e) {
    ISOException.throwIt(ISO7816.SW_WRONG_DATA);
}
```

- Exemple optimisé :

```
if (index < 0 || index >= array.length) {
    ISOException.throwIt(ISO7816.SW_WRONG_DATA);
} else {
    array[index] = value;
}
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

2. **Limitez l'Utilisation des Exceptions au Strict Minimum** :
   - Utilisez-les uniquement pour des erreurs imprévues ou critiques.

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

3. **Utilisation des Exceptions Spécifiques de Java Card** :
   - Les exceptions comme `CryptoException`, `APDUException` ou `ISOException` permettent de signaler des erreurs spécifiques.
   - Exemple :

```
if (pin.isBlocked()) {
    ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
}
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

B. Optimisation du Code

1. **Élimination des Instructions Inutiles** :
   - Identifiez et supprimez les opérations redondantes.
   - Exemple inefficace :

```
for (short i = 0; i < array.length; i++) {
    if (array[i] == 0) {
        continue; // Redondant
    }
}
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

- Exemple optimisé :

```
for (short i = 0; i < array.length; i++) {
    if (array[i] != 0) {
        // Traitement ici
    }
}
```

Bonnes Pratiques de Développement Sécurisé dans Java Card

2. **Utilisation des Méthodes Non Atomiques** :
   - Lorsque les garanties transactionnelles ne sont pas nécessaires, utilisez les méthodes non atomiques (plus rapides).
   - Exemple :

```
Util.arrayCopyNonAtomic(src, (short) 0, dest, (short) 0, length);
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

3. **Simplification des Boucles** :
   - Évitez les calculs inutiles dans les boucles.
   - Exemple inefficace :

```
for (short i = 0; i < array.length; i++) {
    process(array[i]);
    short size = array.length; // Répété inutilement
}
```

   - Exemple optimisé :

```
short size = array.length;
for (short i = 0; i < size; i++) {
    process(array[i]);
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

C. Minimisation de la Taille du Code

1. **Modularité** :
   - Divisez les fonctionnalités en méthodes réutilisables.
   - Exemple :

```
private void updateBalance(short amount) {
    balance += amount;
}
```

Utopios® Tous droits réservés

Bonnes Pratiques de Développement Sécurisé dans Java Card

2. **Facteurs Constants** :
   - Définissez des constantes au lieu d'utiliser des valeurs magiques dans le code.
   - Exemple :

```
private static final byte PIN_MAX_TRIES = 3;
```

Utopios® Tous droits réservés

Résumé des Bonnes Pratiques

Aspect	Bonne Pratique
Mémoire	Utiliser la RAM pour les données temporaires, limiter les écritures en EEPROM.
Réutilisation	Réutiliser les objets pour éviter la fragmentation.
Exceptions	Éviter les exceptions inutiles, gérer les erreurs avec des validations explicites.
Code	Simplifier les boucles, éviter les instructions inutiles, minimiser la taille.
Performances	Privilégier les méthodes non atomiques si possible, optimiser les structures.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

GlobalPlatform est un standard essentiel pour la gestion des applications et des mécanismes de sécurité sur les cartes à puce.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

1. Présentation de GlobalPlatform

GlobalPlatform est un standard mondial pour :

1. **Gérer les applications sur une carte** :
   - Installer, mettre à jour, supprimer et personnaliser les applications.
   - Contrôler les accès aux données sensibles.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

2. **Assurer l'interopérabilité** :
   - Les applications conformes peuvent fonctionner sur toutes les cartes suivant ce standard.
3. **Protéger les données** :
   - Via des canaux sécurisés et des domaines de sécurité.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

B. Mécanismes de Sécurité de GlobalPlatform

1. Canaux Sécurisés (Secure Channels)

Un **canal sécurisé** chiffre et authentifie les communications entre un terminal ou un serveur et la carte.

- **Étape 1 : Initialisation du canal sécurisé** :
  Utilisation des clés de sécurité partagées pour établir le canal.

```
gp -open -key 404142434445464748494A4B4C4D4E4F
```

  - `-key` : Clé maître utilisée pour authentifier l'accès.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

- **Étape 2 : Envoi d'une commande via un canal sécurisé** :
  Une commande `INSTALL` pour charger un fichier CAP :

```
gp -install MyApplet.cap -sc
```

  - `-sc` : Indique que le canal sécurisé est utilisé.
  - Le fichier est chiffré pendant le transfert.

Résultat : La commande est protégée contre toute interception ou modification.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

2. **Isolation avec les Domaines de Sécurité**

Chaque entité (banque, opérateur télécom, etc.) a son **domaine de sécurité** :

- Chaque domaine est isolé des autres.
- Les données et les applications d'un domaine sont inaccessibles depuis d'autres domaines sauf autorisation explicite.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

Exemple d'isolation :

- **Domaine 1** (banque) contient une applet bancaire.
- **Domaine 2** (opérateur télécom) contient une applet de gestion SIM.

Si l'applet bancaire tente d'accéder aux données de l'applet SIM sans autorisation :

```
ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
```

Résultat : L'accès est refusé par le firewall.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

3. Cycle de Vie des Entités

GlobalPlatform définit un **cycle de vie** clair pour les applications et la carte.

Exemple de cycle de vie d'une application :

1. **INSTALL** :
   - L'application est chargée dans la carte mais pas encore activée.

```
gp -install MyApplet.cap
```

GlobalPlatform et Gestion des Domaines de Sécurité

2. **PERSONALIZE** :
   - L'application est configurée avec des données spécifiques (ex. un PIN par défaut).

```
pin.update(defaultPIN, (short) 0, (byte) defaultPIN.length);
```

3. **ACTIVATE** :
   - L'application devient fonctionnelle.

```
gp -activate
```

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

4. **DELETE** :
   - L'application est supprimée.

```
gp -delete MyApplet
```

GlobalPlatform et Gestion des Domaines de Sécurité

2. Avantages de GlobalPlatform

A. Création et Gestion des Domaines de Sécurité

Un domaine de sécurité est une zone isolée dans la carte, protégée par des clés cryptographiques. Il est utilisé pour gérer les applications d'une entité spécifique.

Exemple de création d'un domaine de sécurité :

- **Commande pour créer un domaine :**

```
gp -create SecurityDomain.cap
```

  - Résultat : Un nouveau domaine est créé avec ses propres clés.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

- **Gestion des clés du domaine :**
  - Chaque domaine a des clés uniques :

```
AESKey domainKey = (AESKey) KeyBuilder.buildKey(KeyBuilder.TYPE_AES, KeyBuilder.LENGTH_AES_128, false);
domainKey.setKey(keyData, (short) 0);
```

2. **Partage Sécurisé entre Domaines :**

Les objets ou données peuvent être partagés entre deux domaines si explicitement autorisés.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

Exemple :

- **Domaine 1 (banque)** partage des données avec **Domaine 2 (fidélité)** :
  - L'applet bancaire expose un objet partagé via `Shareable` :

```
public Shareable getShareableInterfaceObject(AID clientAID, byte parameter) {
    if (clientAID.equals(trustedClientAID)) {
        return sharedObject;
    }
    return null;
}
```

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

- **Domaine 2** utilise cet objet :

```
Shareable sharedObject = JCSystem.getAppletShareableInterfaceObject(serverAID, (byte) 0);
```

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

B. Utilisation dans le Développement d'Applications Java Card

1. **Interopérabilité :**
    - Les applications développées selon les standards GlobalPlatform fonctionnent sur toutes les cartes conformes.

Exemple de compatibilité :

- Une applet bancaire développée pour une carte X peut être utilisée sur une carte Y tant qu'elle est conforme à GlobalPlatform.

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

2. Gestion Dynamique des Applications :

- **Installation d'une applet Java Card :**

```
gp -install MyApplet.cap -key MyKey
```

- **Mise à jour d'une applet :**
  Une nouvelle version est installée sans affecter les données utilisateur existantes.

```
gp -update MyAppletNewVersion.cap
```

Utopios® Tous droits réservés

GlobalPlatform et Gestion des Domaines de Sécurité

3. **Partage d'objets sécurisé :**
    - Les API standardisées de GlobalPlatform simplifient les interactions entre applets.

Exemple de collaboration :

- Une applet de paiement partage un solde avec une applet de fidélité pour calculer des points de récompense.

Utopios® Tous droits réservés

UTOPIOS®
TRANSFORMATION DIGITALE INTELLIGENTE

Merci pour votre attention

Des questions ?

Utopios® Tous droits réservés