TP2 : Interaction avec la carte à puce à l'aide d'un programme Java sous Windows

Samia BOUZEFRANE
http://cedric.cnam.fr/~bouzefra/pfsem10-11.html
samia.bouzefrane@cnam.fr

Laboratoire CEDRIC
Chaire Systèmes Embarqués et Enfouis
CNAM

1. API JSR 268

JSR 268 définit une API Java pour permettre la communication avec les cartes à puce en utilisant le protocole ISO 7816-4. Il permet ainsi aux applications écrites en Java 1.6 d'interagir avec des applications qui tournent sur la carte à puce, en utilisant Linux ou Windows.

Description du Package javax.smartcardio

Classe et Constructeur	Méthodes de la classe
ATR : Réponse à un Reset Answer To Reset **ATR**(byte[] atr) construit un ATR à partir d'un tableau d'octets.	boolean **equals**(Object obj) Compares the specified object with this ATR for equality. byte[]**getBytes**() Returns a copy of the bytes in this ATR. String **toString**() Returns a string representation of this ATR. …
Card : Etablissement de connexions Protected **Card**() construit un objet Card.	abstract void **disconnect**(boolean reset) déconnexion de la carte. abstract ATR **getATR**() retourne l'ATR de la carte. abstract String **getProtocol**() retourne le protocole de cette carte. abstract CardChannel **getBasicChannel**() retourne le CardChannel d'un canal logique de base…

CardChannel : Connexion logique à la carte à puce
 	
`protected` **`CardChannel`**`()`
construit un objet CardChannel
	`abstract void` **`close`**`()`
	ferme le CardChannel
	
	`abstract ResponseAPDU` **`transmit`**`(CommandAPDU command)`
	envoie la commande APDU vers la carte et retourne une réponse APDU.
	
	…

CardTerminal :**Création d'un lecteur de carte**
 	
`protected` **`CardTerminal`**`(`TerminalFactory factory) construit un nouvel objet CardTerminal
	`abstract Card` **`connect`**`(String protocol)`
	établit une connexion avec la carte.
	
	`abstract String` **`getName`**`()`
	retourne le nom du lecteur.
	
	`abstract boolean` **`isCardPresent`**`()` retourne vrai si la carte est présente dans le lecteur, faux sinon.
	
	…

CommandAPDU : Une commande APDU tel que éfini dans ISO 7816-4
 	
`CommandAPDU`(byte[] apdu)
construit une commande APDU à partir d'un tableau d'octets.
 	
`CommandAPDU`(byte cla, byte ins, byte p1, byte p2) construit une commande APDU à partir des 4 octets de l'entête.
 	
etc.
	`byte[]` **`getBytes`**`()`
	retourne une copie des octets de l'APDU.
	
	`byte` **`getCLA`**`()`
	retourne la valeur de CLA.
	
	`byte` **`getINS`**`()`
	retourne la valeur en octet de INS.
	
	`byte` **`getP1`**`()`
	retourne la valeur en octet du paramètre P1.
	
	`byte` **`getP2`**`()`
	retourne la valeur en octet du paramètre P2.
	
	`int getNe``()`
	retourne le nombre maximal d'octets attendus en réponse.
	
	etc.

ResponseAPDU : la réponse APDU tel que défini dans ISO 7816-4 **`ResponseAPDU`**(byte [] apdu) construit une réponse APDU à partir d'un vecteur d'octets contenant l'APDU complète (données et SW1 et SW2).	**`byte [] getBytes()`** retourne une copie de l'APDU en octets. **`byte [] getData()`** retourne une copie en octets des données envoyées en réponse à la commande APDU. **`byte getSW1()`** retourne la valeur en octet de l'état SW1. **`byte getSW2()`** retourne la valeur en octet de l'état SW2. etc.
TerminalFactory : une fabrique pour les objets `CardTerminal`	`CardTerminal` **`getTerminal`**`(String name)` retourne le terminal avec le nom spécifié ou null si le terminal spécifié n'existe pas. `static TerminalFactory` **`getDefault`**`()` retourne une instance de TerminalFactory par défaut. etc.

II. Outils de développement

II.1. Outils matériels

Un lecteur de carte SIM et une carte SIM (le pilote du lecteur ayant été installé durant le 1er TP).

II.1. Outils logiciels nécessaires

a) Télécharger et installer JDK 1.6 (utiliser le lien suivant pour le télécharger : (http://java.sun.com/javase/downloads/widget/jdk6.jsp).

b) Télécharger et installer Eclipse galileo pour Windows (http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/galileo/SR1/eclipse-java-galileo-SR1-win32.zip).

c) Télécharger l'outil de développement de JavaCard JSR268TK.zip dans un répertoire local par exemple C:\JavaCard, à partir du lien suivant :
http://cedric.cnam.fr/~bouzefra/cours/cours_SEM/JSR268TK.rar

d) Pour connaître, le nom du lecteur de carte, il faut consulter la base des registres (Sous Windows, cliquer sur le menu **Démarrer** et choisir **Exécuter** pour exécuter la commande **regedit** et parcourir le chemin suivant):
Hkey local machine/software / Microsoft/cryptography/calais/readers

III. Préparation de l'environnement de développement

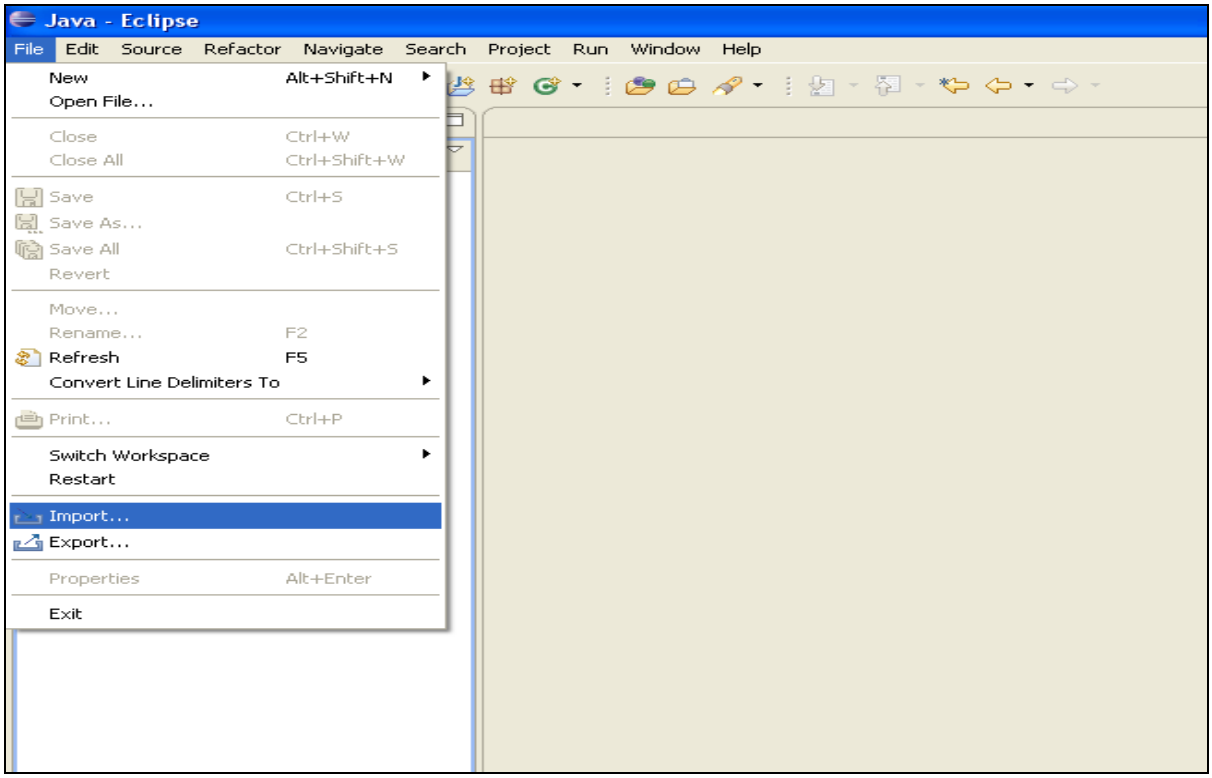
III.1 importer l'API JSR268 de développement sous éclipse :

a) Décompresser le projet c:\JavaCard\JSR268TK.zip dans le même répertoire.
b) Lancer Eclipse.
c) Importer le projet java C:\JavaCard\JSR268TK\JSR268TK\JSR268TK-2 de la manière suivante :

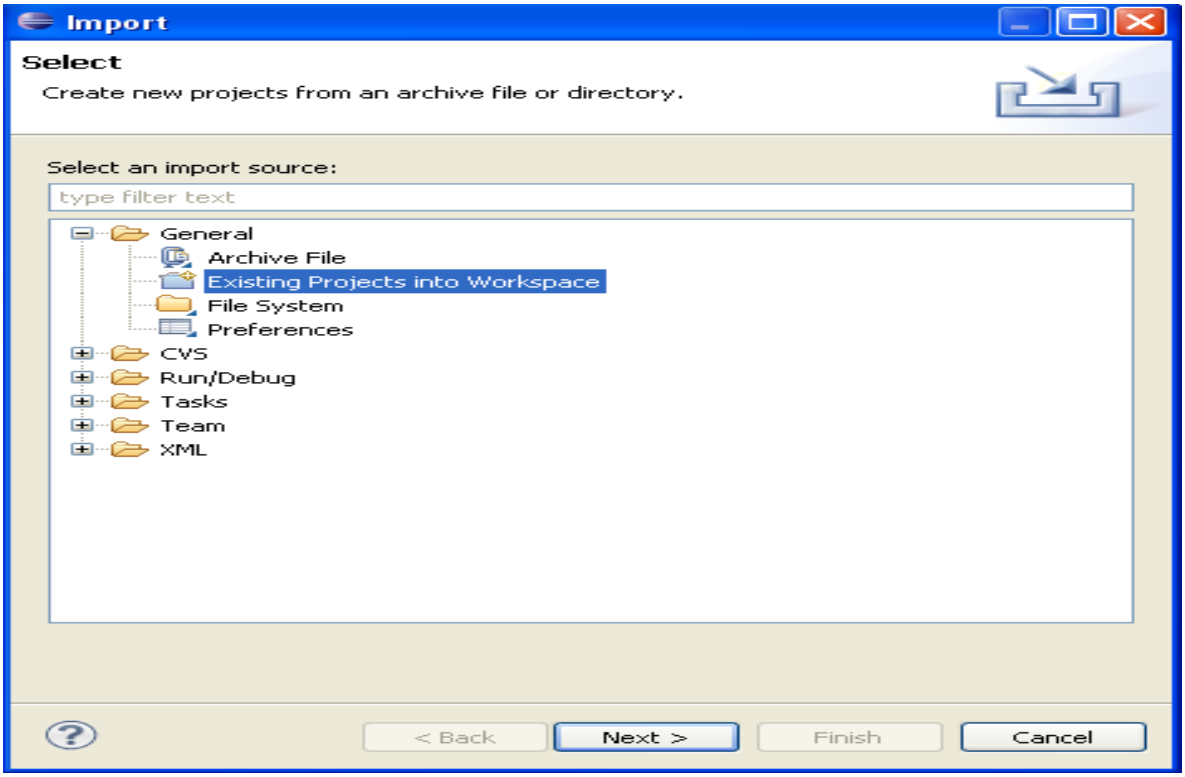
File → Import → General → Existing Project into WorkSpace → Browse

Chercher l'emplacement où vous avez décompressé le ficher JSR268TK.zip. Dans notre cas c'est : C:\JavaCard\JSR268TK\JSR268TK-2. Sélectionner le répertoire JSR268TK-2 et puis faire OK.

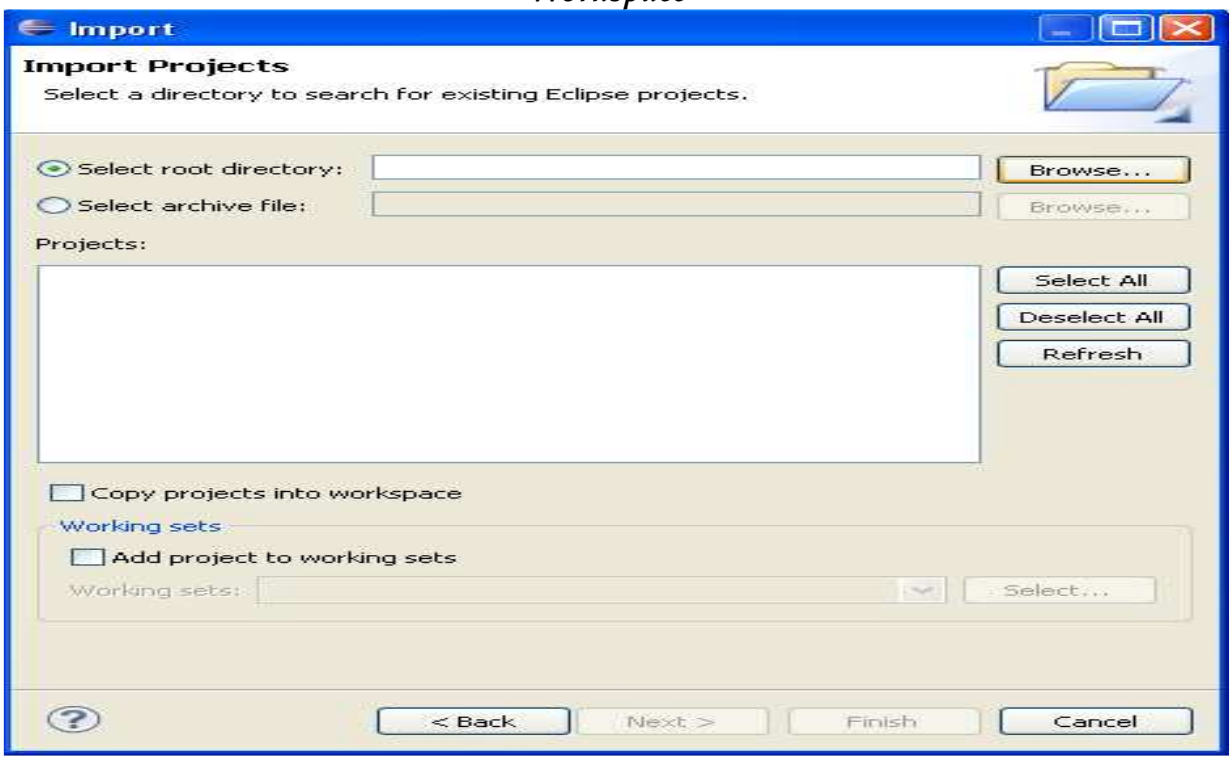
Les captures d'écran suivantes explicitent cette manipulation.



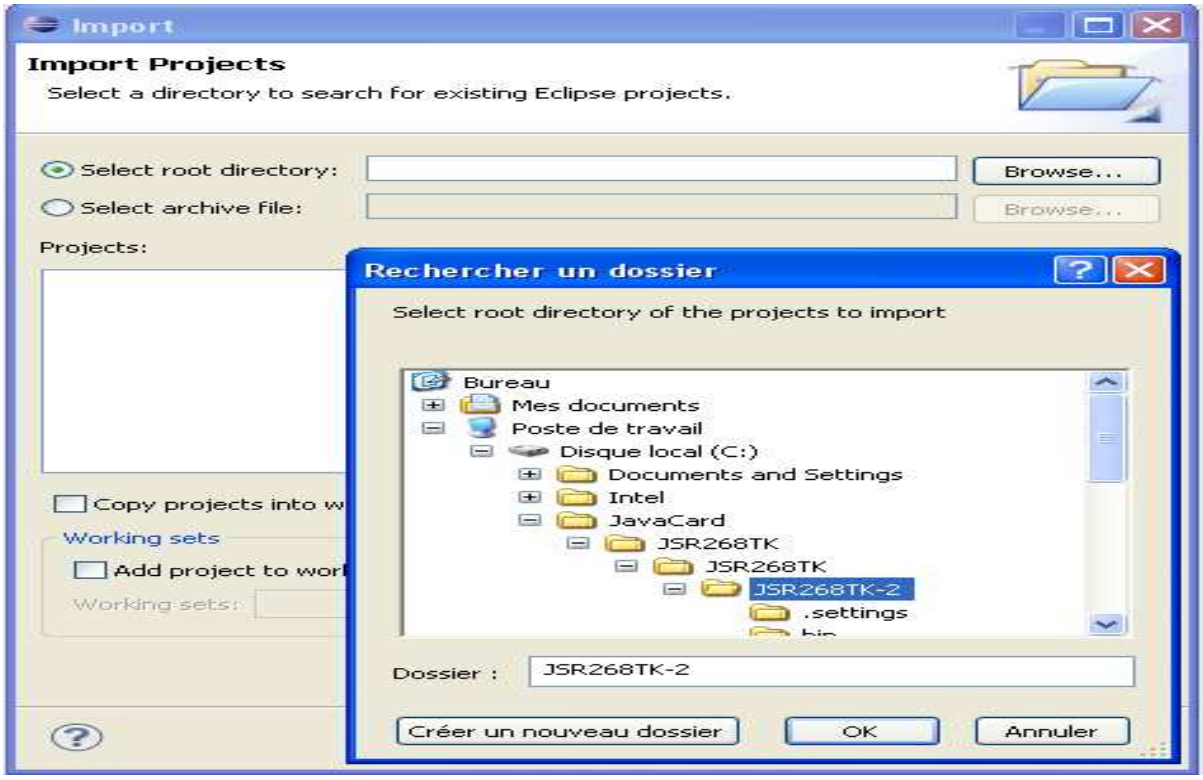
Cliquer sur le bouton ***File*** ensuite sur ***Import***



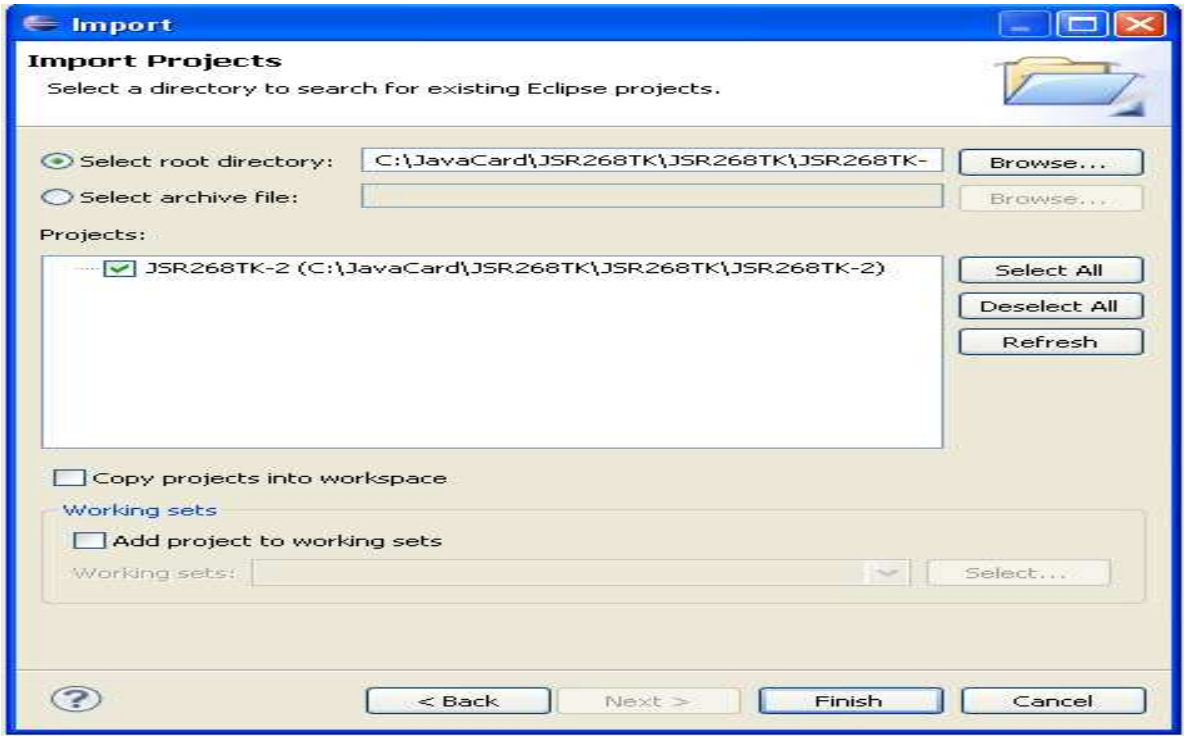
Cliquer alors sur le bouton « + » devant ***General*** → Cliquer sur ***Existing Projects into Workspace***



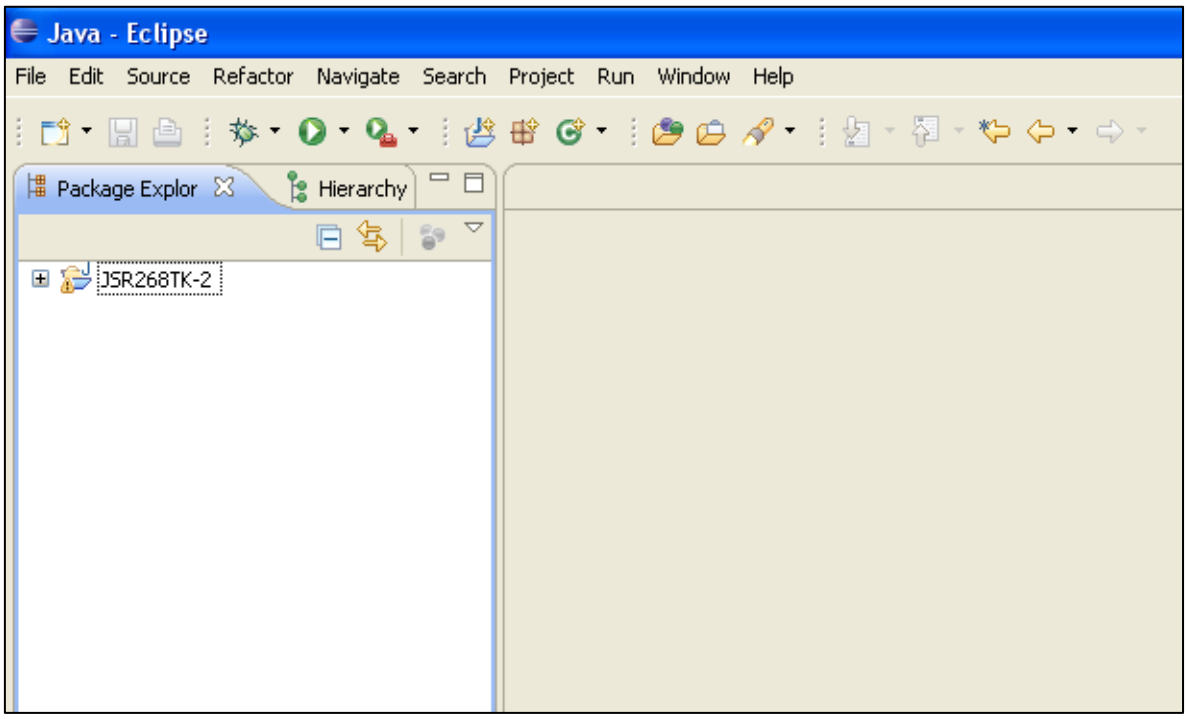
Cliquer alors sur le bouton ***Browse*** pour rechercher le projet à importer.



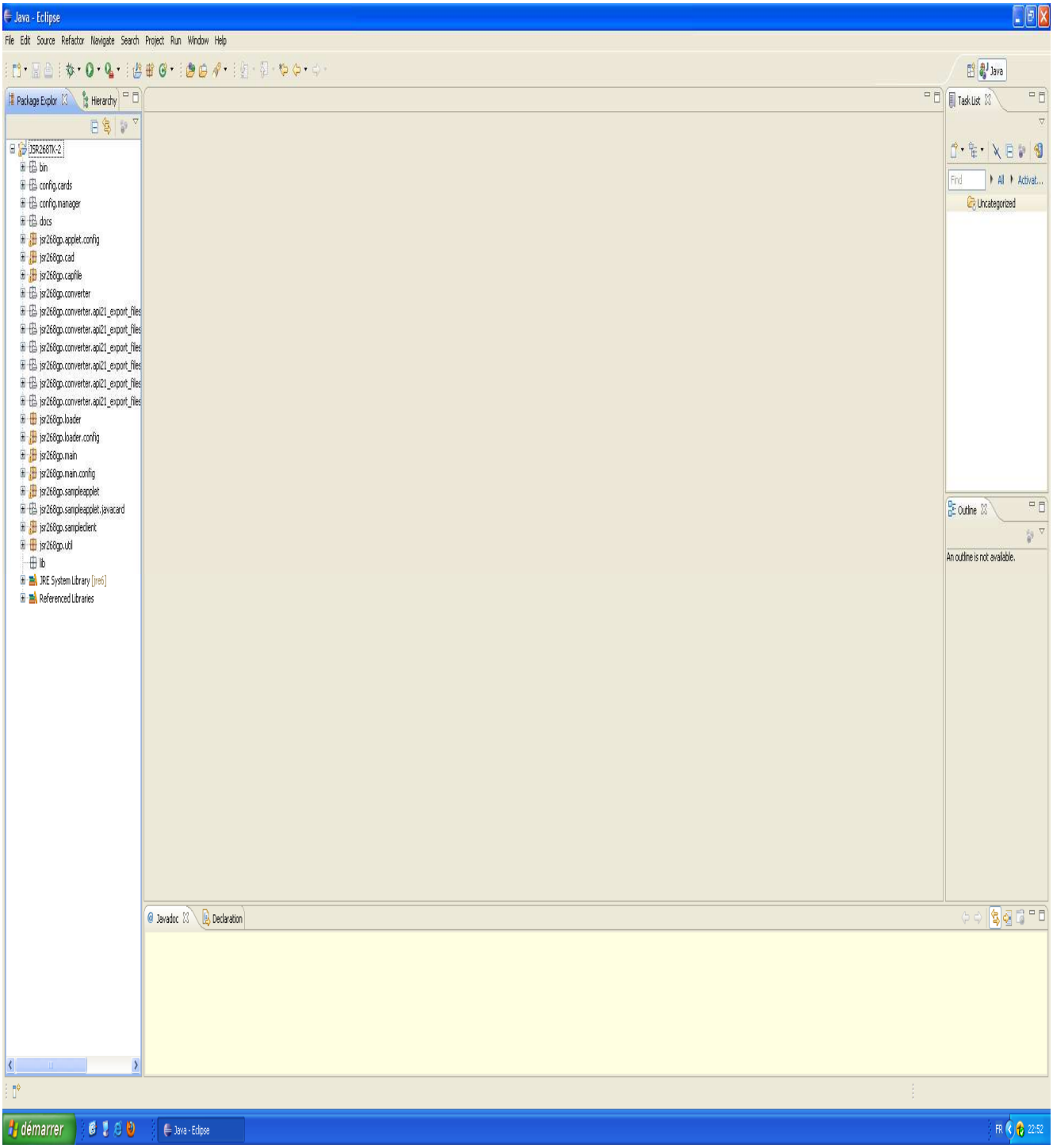
Cliquer alors sur le bouton + devant ***C*** → Cliquer sur + JavaCard → cliquer sur + JSR268TK → Cliquer sur + JSR268TK →Cliquer sur JSR268TK-2 → Cliquer sur le bouton ***OK***



Cliquer sur le bouton ***Finish***

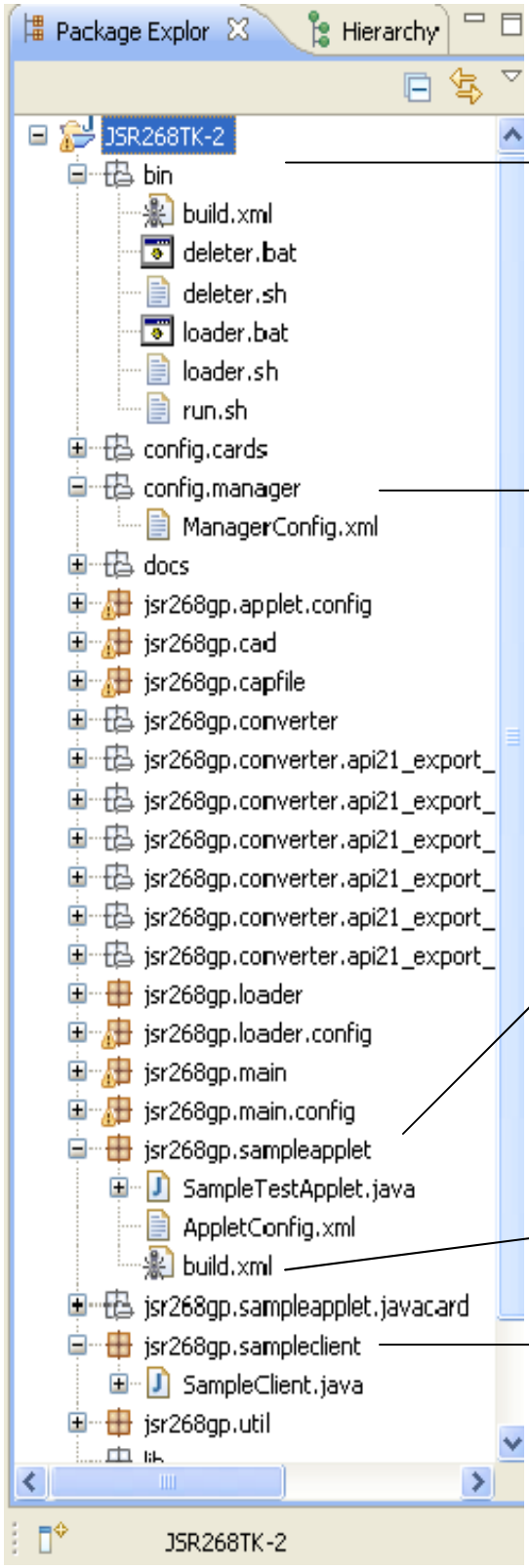


Vous devriez voir cette fenêtre, mais afin de visualiser les packages de ce projet cliquer sur « + » devant JSR268TK-2



Vous devriez voir cette fenêtre

Ci-dessous un schéma qui décrit d'une manière générale les différentes parties de cet API. Dans le TP sur Java Card, cet environnement nous permettra de développer une application Java Card (côté client et côté carte).



build.xml : comporte l'exécution de trois programmes
deleter : *supprimer* l'ancienne applet sur la carte.
loader : charger la nouvelle applet sur la carte.
run : exécuter le programme du terminal.

ManagerConfig.xml : sert à configurer le lecteur de carte à puce en mettant son nom.

SampleTestApplet.java : le squelette de l'applet

Build.xml : sert à compiler SampleTestApplet.java et générer éventuellement le fichier SampleTestApplet.cap qui sera installé sur la carte. Ce fichier comprend bien entendu l'AID de l'applet ainsi que l'AID du package auquel l'applet appartient.

SampleClient.java : la partie du terminal qui va interroger la carte.

III.2 Configuration de l'environnement de développement :

a) Modification du ManagerConfig.xml en changeant le nom du lecteur.
b) Développement de la partie terminal (SampleClient.java).
c) Exécution de la partie terminal.

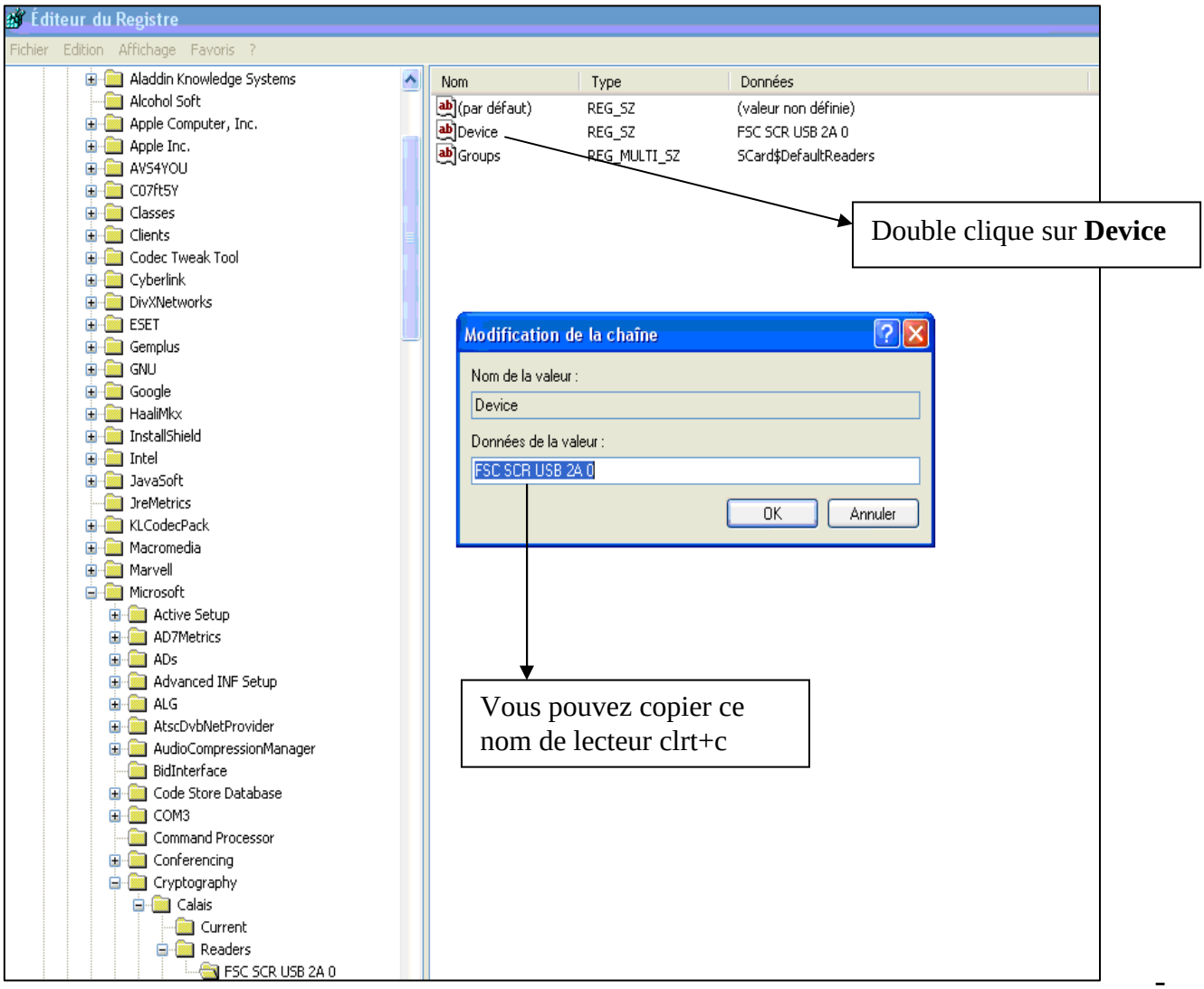
a) Modification du *ManagerConfig.xml*

- La première étape consiste à modifier le fichier ***ManagerConfig.xml*** en mettant le nom de notre lecteur. Ce dernier se trouve dans la base de registres. Nous l'obtenons de la manière suivante :

Bouton Démarrer → ***Exécuter*** → Taper: ***regedit***

Le nom de lecteur se trouve dans l'emplacement suivant:

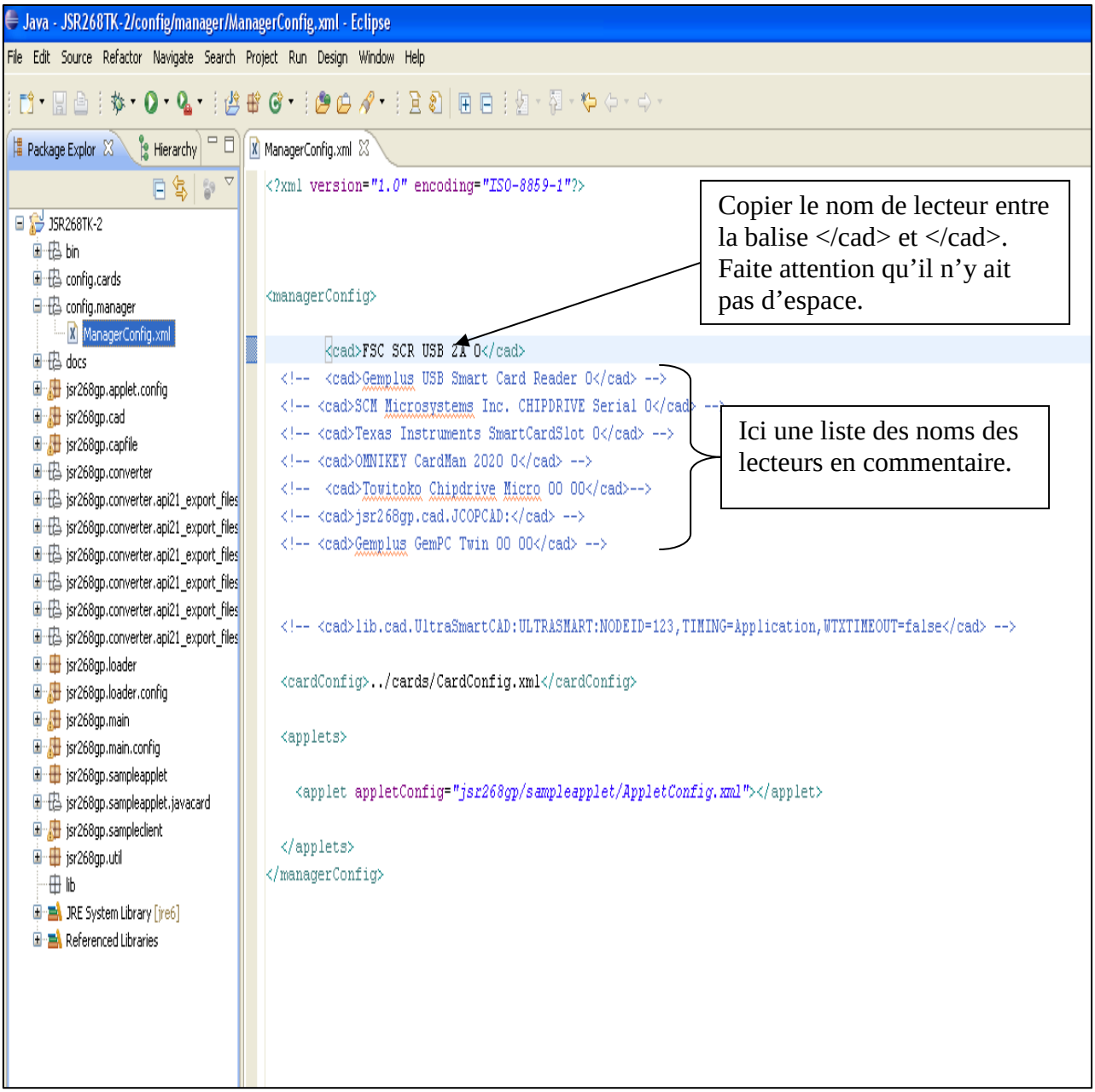
Cliquer sur le bouton + devant ***Hkey local machine*** → ***+ software*** → + ***Microsoft*** → ***+ cryptography*** → ***+ calais*** → ***+ readers***



-

Ouvrir le fichier ManagerConfig.xml (double clique de souris sur le fichier).

- Enregistrer les modifications en tapant Ctrl+S.

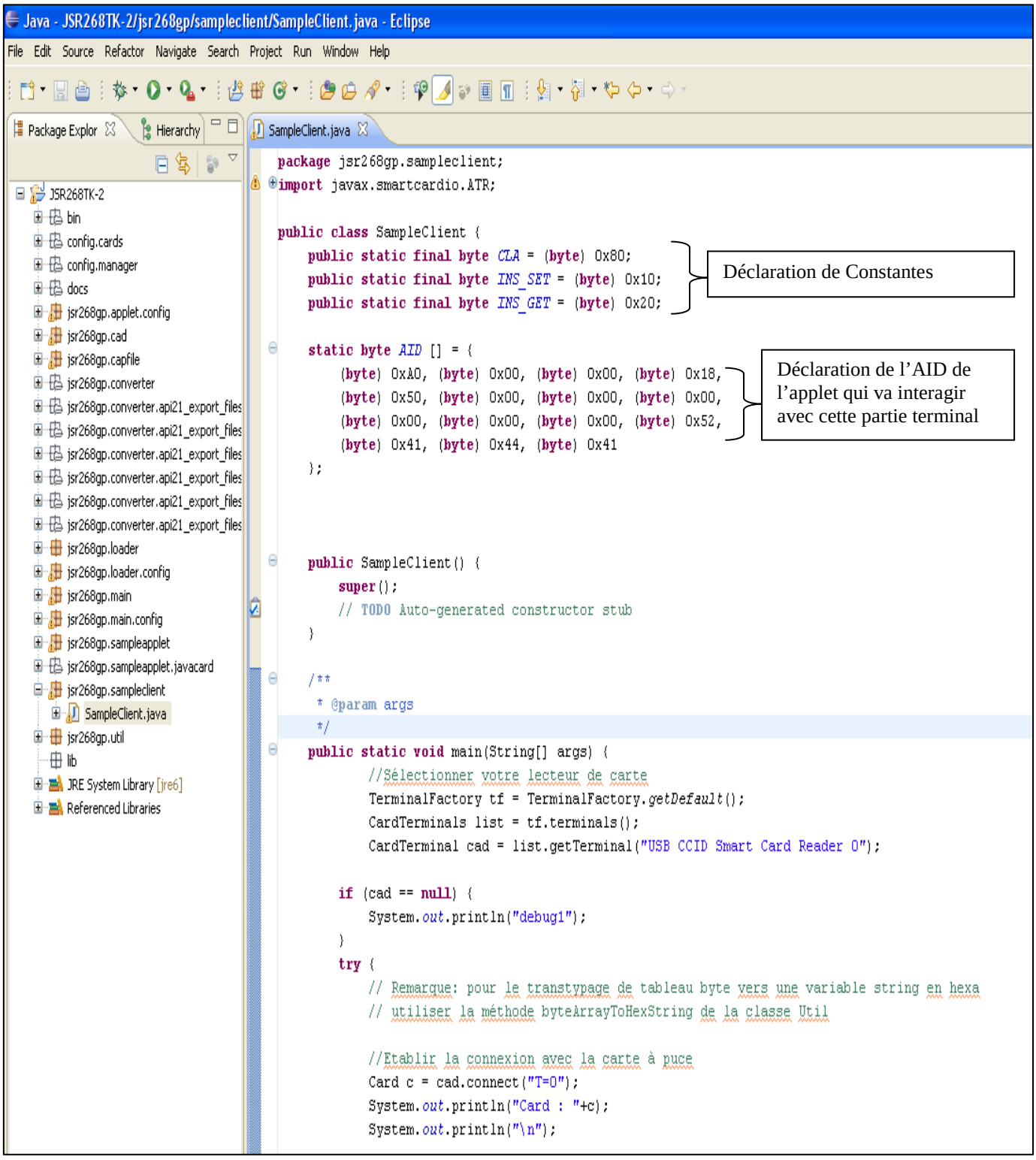


b) Développement de la partie terminal :

-La partie terminal correspond au fichier `SampleClient.java`.

-Ouvrir le fichier `SampleClient.java` (double clique de souris sur le fichier).

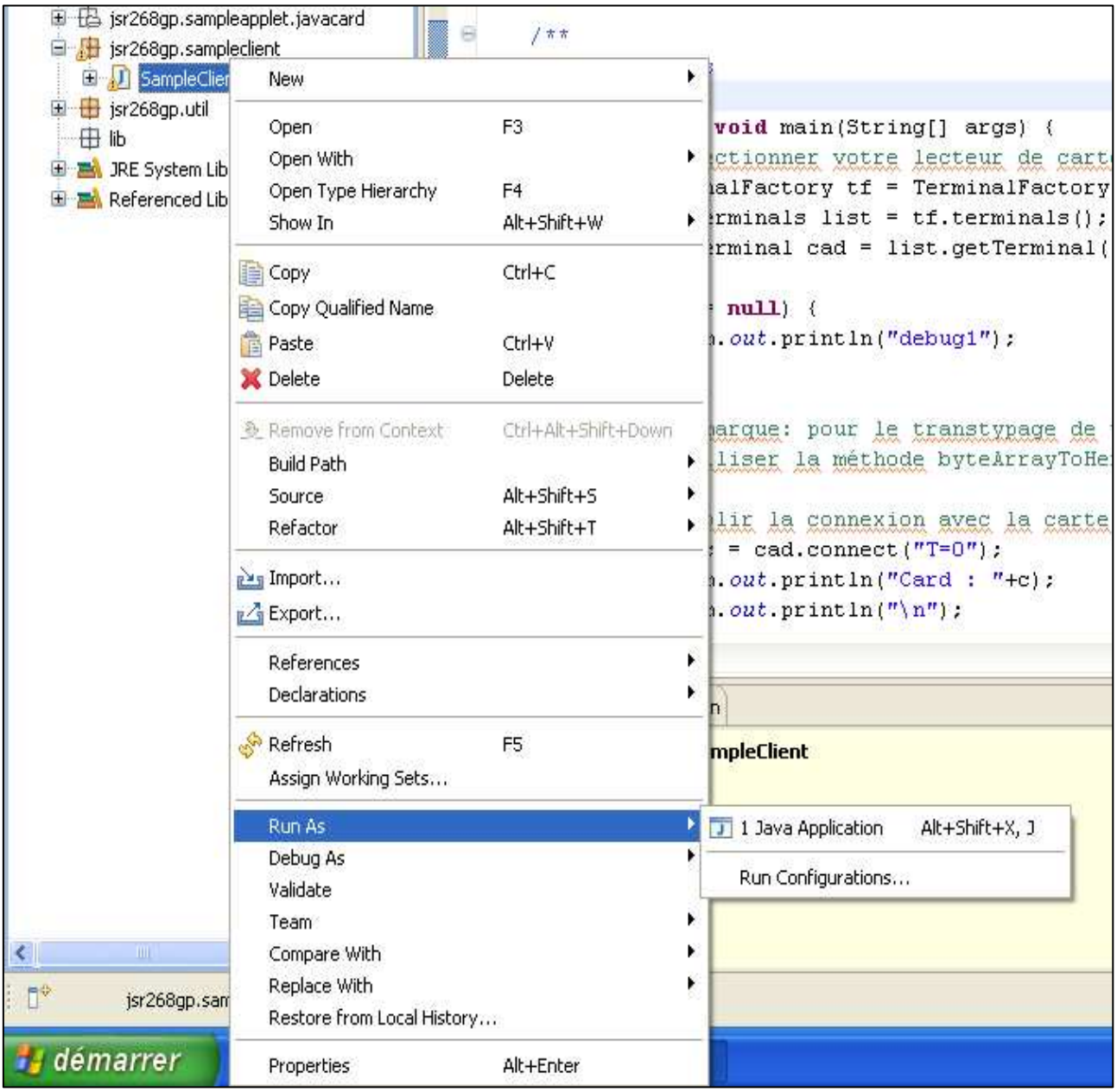
L'image ci-dessous s'affiche :



- Exécuter la partie terminal. Comme dans ce TP, le terminal n'interagit pas avec une applet sur la carte, la déclaration de l'AID ne sera pas utile.

Cliquer sur « + » devant `jsr268.sampleClient`.
Cliquer sur le bouton droit de la souris sur le fichier `SampleClient.java`.
Choisir ***Run As.*** Cliquer ensuite sur ***Java Application.***

Une console affiche le résultat ou d'éventuelles erreurs.



Exercice:

En vous basant sur cet environnement, écrire un programme qui affiche l'ATR de la carte SIM, et qui envoie des commandes APDU pour récupérer quelques informations telles que l'IMSI, TMSI et le fichier des SMS.