Compte Rendu TP1

Préparation de l'Environnement de développement JavaCard

Dhia Ben Hamouda

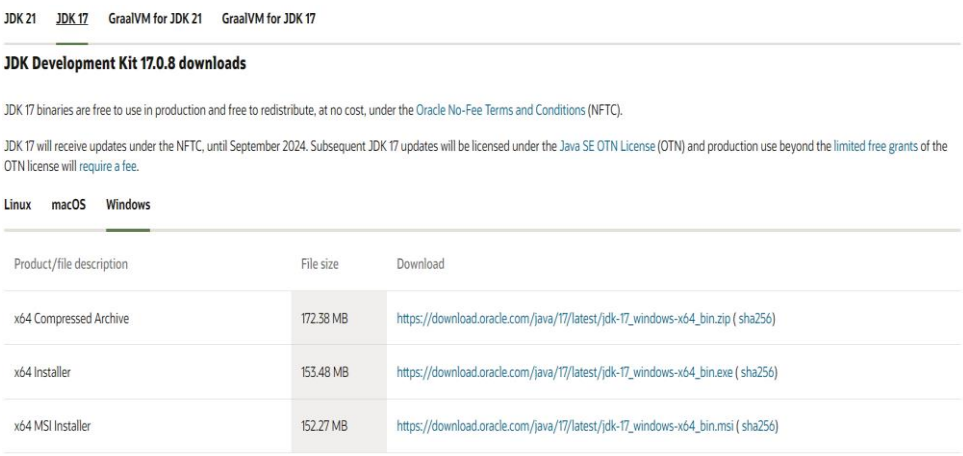
ING INFO 1 TD1

dhiabenhamouda2002@gmail.com

Table des matières

1) *Installation du JDK*

Accéder au cite https://www.oracle.com/java/technologies/downloads/



JDK 21 JDK 17 GraalVM for JDK 21 GraalVM for JDK 17

JDK Development Kit 17.0.8 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 17 will receive updates under the NFTC, until September 2024. Subsequent JDK 17 updates will be licensed under the Java SE OTN License (OTN) and production use beyond the limited free grants of the OTN license will require a fee.

Linux macOS Windows

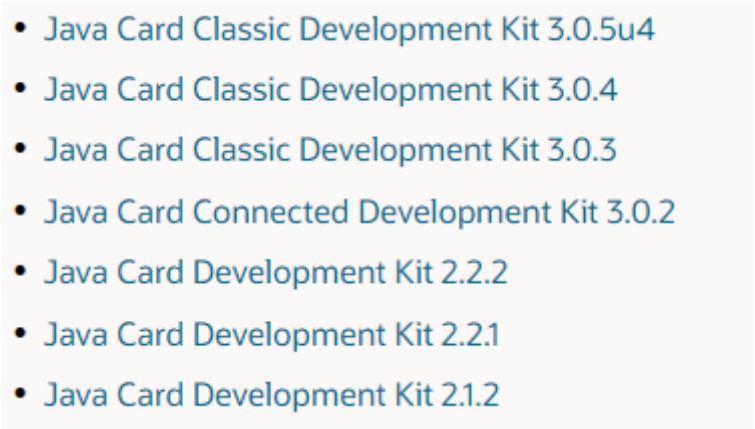
Product/file description	File size	Download
x64 Compressed Archive	172.38 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip (sha256)
x64 Installer	153.48 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe (sha256)
x64 MSI Installer	152.27 MB	https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi (sha256)

On Télécharge le JDK et on l'installe sur la machine.

2) Setup du JCDK:

A) Accéder au site https://www.oracle.com/java/technologies/javacard-sdk-downloads.html

B) On choisie la version voulu à opérer (v2.x | v3.x) et on la télécharge



- Java Card Classic Development Kit 3.0.5u4
- Java Card Classic Development Kit 3.0.4
- Java Card Classic Development Kit 3.0.3
- Java Card Connected Development Kit 3.0.2
- Java Card Development Kit 2.2.2
- Java Card Development Kit 2.2.1
- Java Card Development Kit 2.1.2

C) On décompresse le fichier télécharger dans le répertoire C:\JavaCard

3) Ajouter les variables d'environnement nécessaires

On ajoutera dans cette parties deux variables d'environnements

- JAVA_HOME

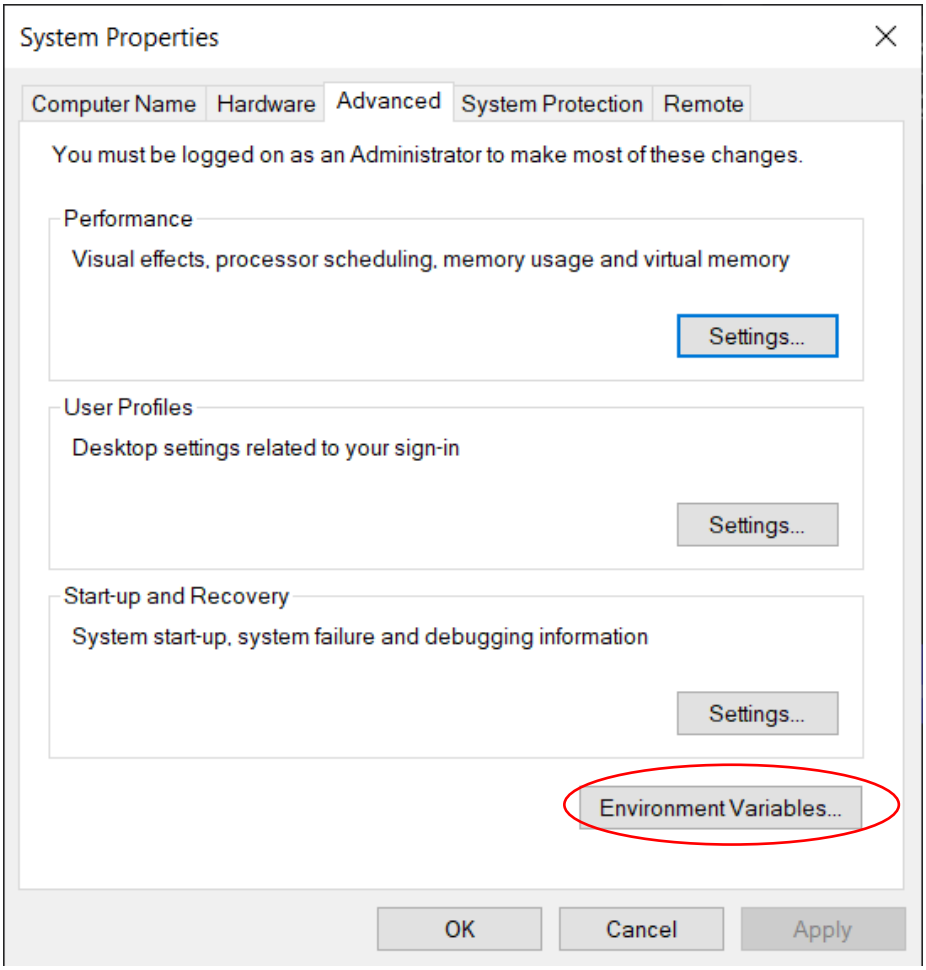
-JC_HOME

Ceci est pour rendre au system découvrable les exécutables du JDK et du JCRE (JavaCard RunTime Environment)

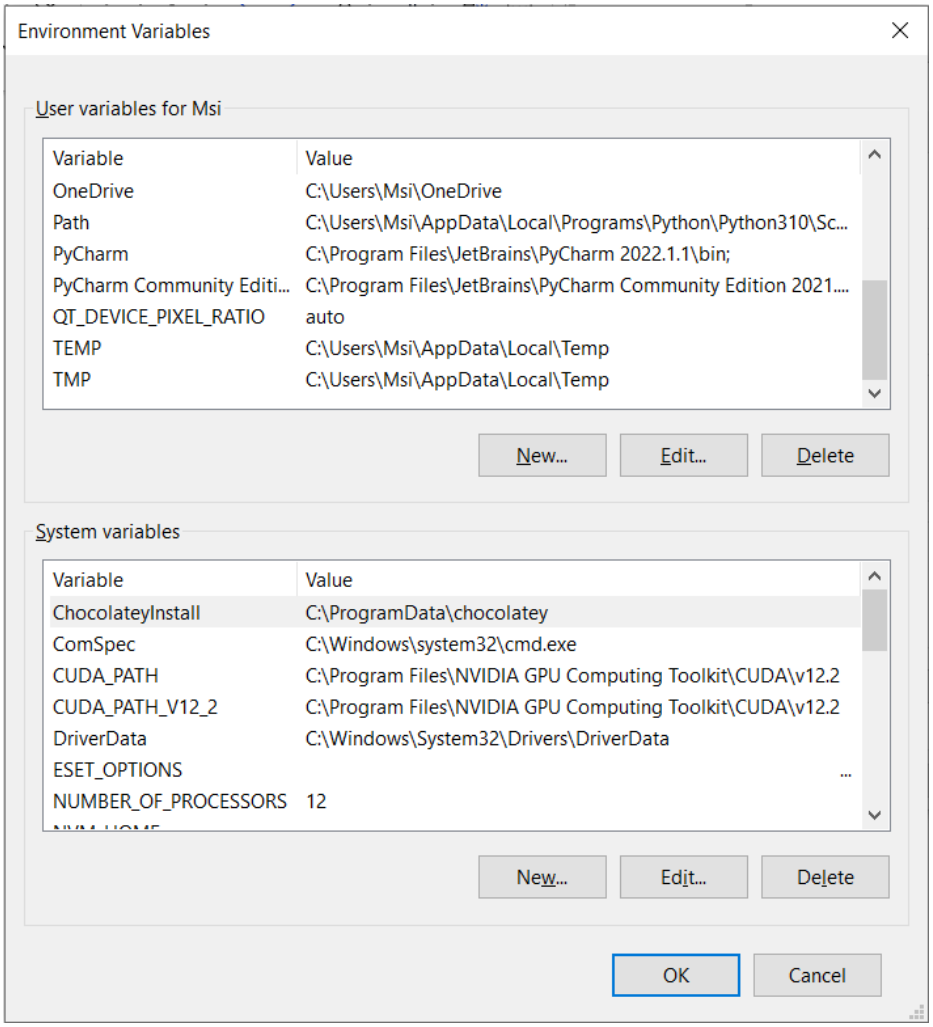
A) JAVA_HOME:

-Accéder au variable d'environnement:

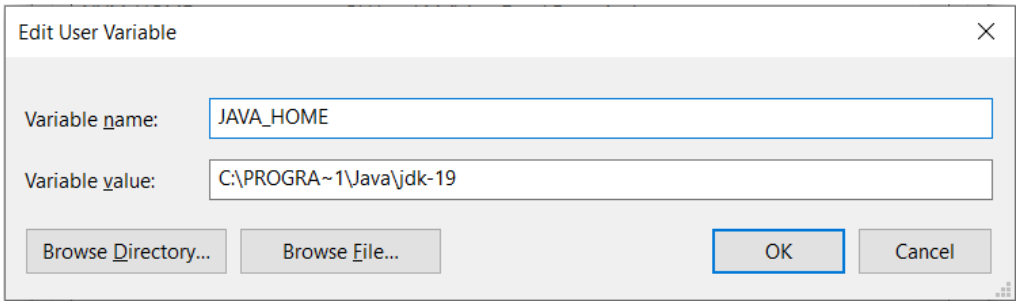
-Appuyer sur variables d'environnement:


System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
You must be logged on as an Administrator to make most of these changes.
Performance
Visual effects, processor scheduling, memory usage and virtual memory
Settings...
User Profiles
Desktop settings related to your sign-in
Settings...
Start-up and Recovery
System start-up, system failure and debugging information
Settings...
Environment Variables...
OK
Cancel
Apply

-Appuyer sur nouveau



-Entrer comme suite en changeant «Variable value» avec le sous-dossier ou vous avez installer votre JDK:



<u>Remarque:</u> la racine du dossier ne doit pas contenir des espaces

PROGRA~1 est juste un autre nom du dossier «Program Files» mais sans espaces

Pour trouver les autres nom de certains dossier system, taper la commande «dir /X» dans le CMD

-Finalement, trouver la variable PATH, appuyez dessus, cliquer sur modifier et ajouter le r %JAVA_HOME%\bin HOME%\bin»

B) JC_HOME

-Répéter les memes étapes pour la variable JAVA_HOME en remplacent le dossier d'installation du JDK par C:\JavaCard

Remarque: Je voudrais travailler avec les version 2 et 3 du javacard. Donc j'ai créer 3 variables d'environnement:

-JC_HOME_v2

-JC_HOME_v3

-JC_HOME

Affin de changer de version, il suffit de changer la valeur de la variable d'environnement «JC_HOME» en «JC_HOME_vX» en changeant le X avec 2 ou 3

4) Vérifier la bonne installation du JCDK

Entrer la commande «java -version» est un message similaire a celui-ci devrait apparaitre:

```
C:\Users\Msi>java -version
java version "20" 2023-03-21
Java(TM) SE Runtime Environment (build 20+36-2344)
Java HotSpot(TM) 64-Bit Server VM (build 20+36-2344, mixed mode, sharing)
```

5) Vérifier la bonne installation du JavaCard

Il suffit d'ouvrir le CMD et taper la commande «apdutool» si le message suivant apparaît:

(v2)

```
C:\Users\Msi>apdutool
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
Opening connection to localhost on port 9025.
java.net.ConnectException: Connection refused: connect
```

(v3)

```
C:\Users\Msi>apdutool
ApduTool [v3.1.0]
    Copyright (c) 1998, 2021, Oracle and/or its affiliates. All rights reserved.


Opening connection to localhost on port 9025.
Connected.
java.net.ConnectException: Connection refused: connect
```

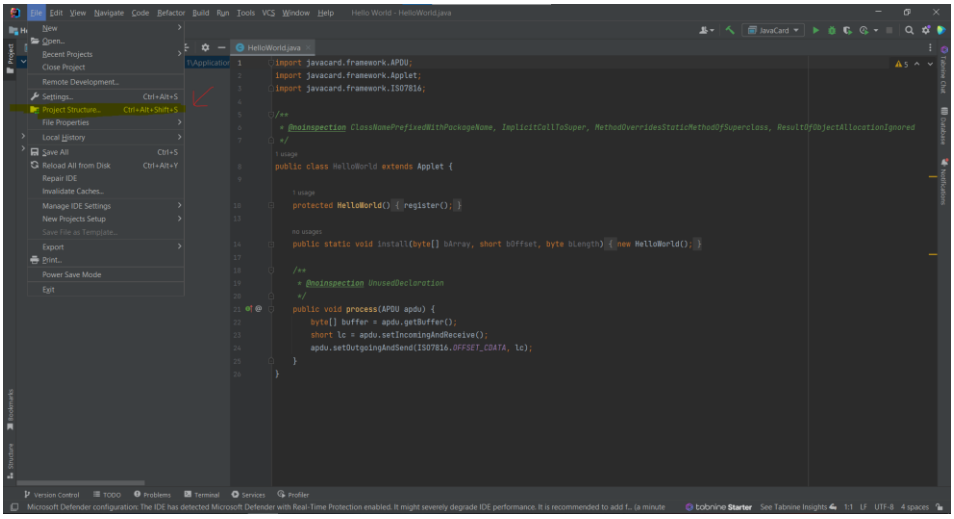
Alors l'installation est réussite et la commande est reconnu par le système

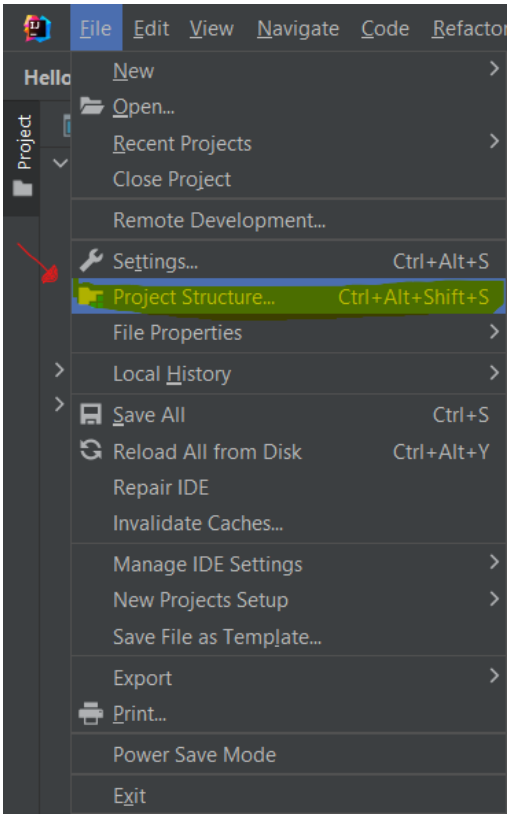
6) Ajouter les librairies(API) JavaCard à votre environnement de développement

Pour les utilisateurs IntelliJ:

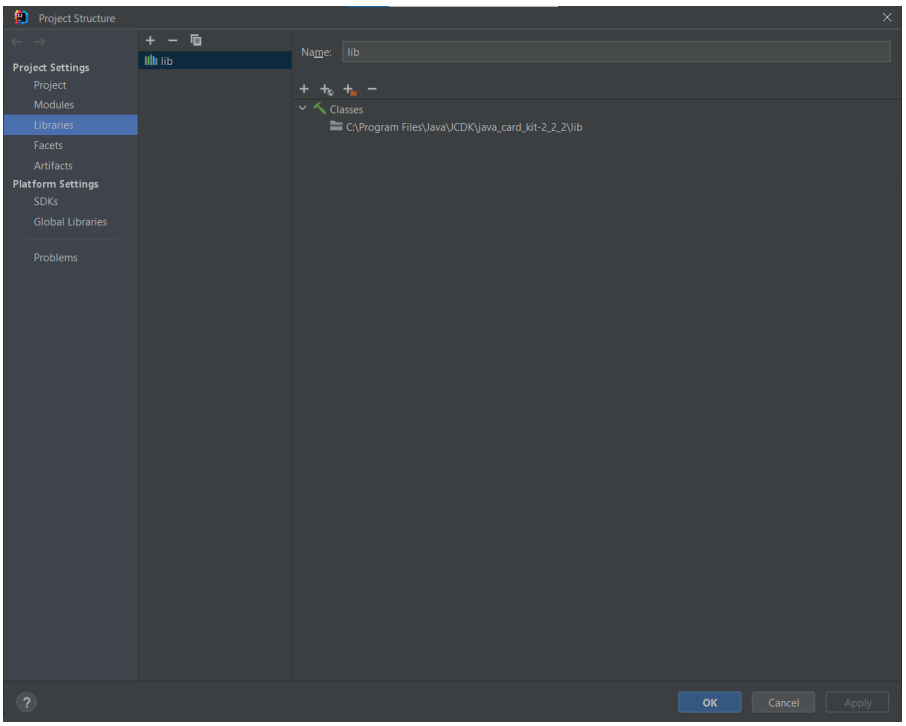
A) Télécharger un template JavaCard de GitHub (lien : https://github.com/FractalizeR/IntelliJ-IDEA-Java-Card-Project-Template.git)

B) Accédez au «File -> Project Structure»





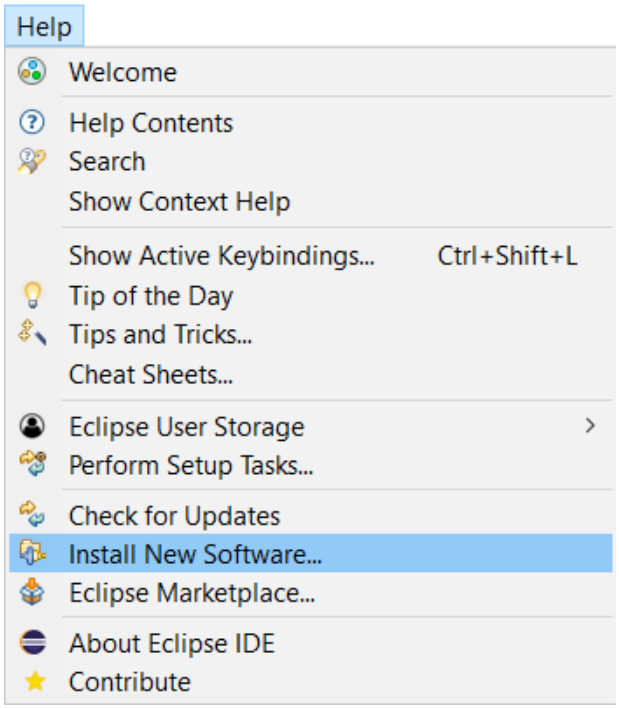
C) Tapez sur le menu Librairies
D) Cliquer sur l'icone +
E) Parcourir le dossier C:\JavaCard
F) Ajouter le sous-dossier «lib»



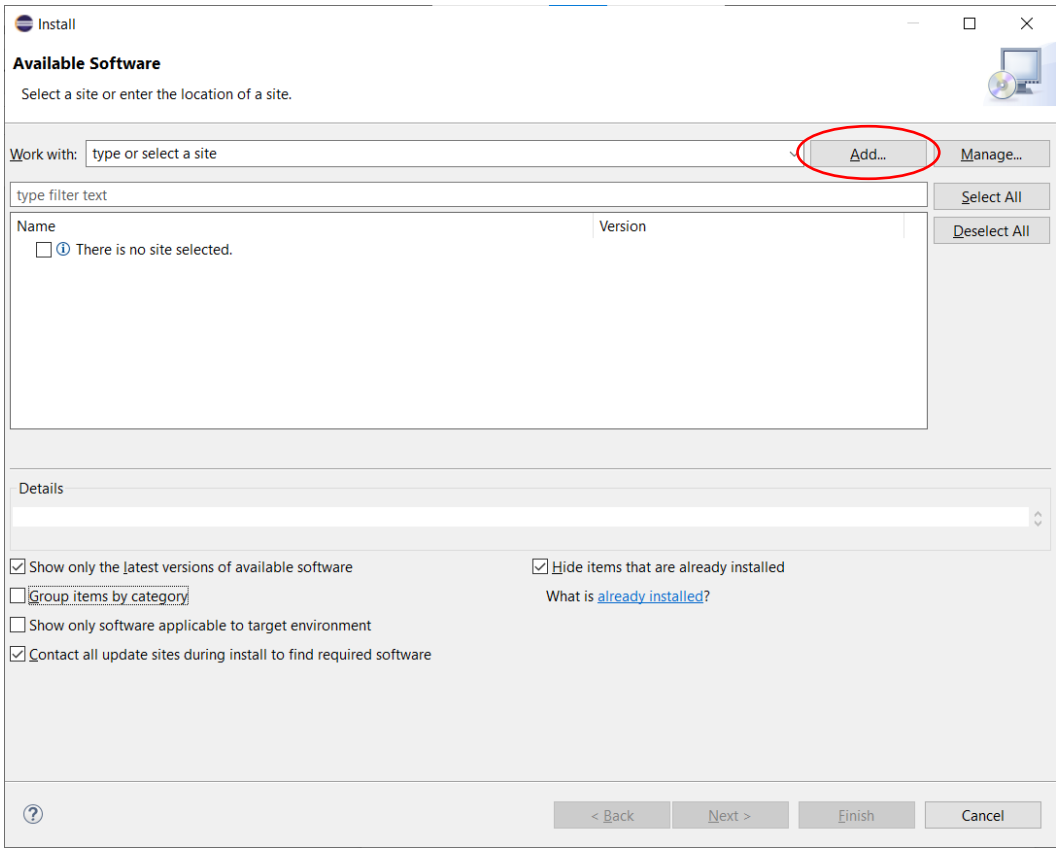
G) Cliquez sur confirmer

Pour les utilisateurs Eclipse (JavaCard v3.x):

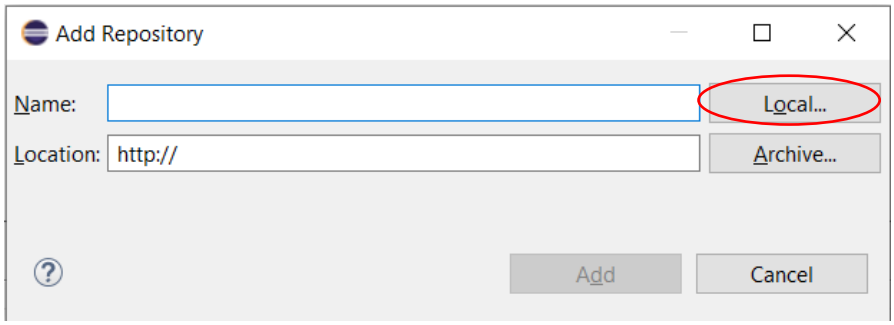
A) Accéder au menu «HELP» et à «Install New Software»



B) Taper sur «Ajouter»



C) Cliquer dur «Archive»

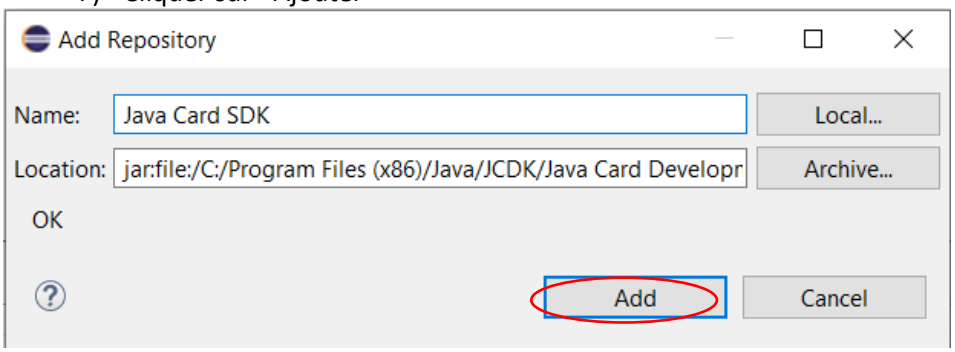


D) Selectionner le kit de développement Eclipse plug-in dans :

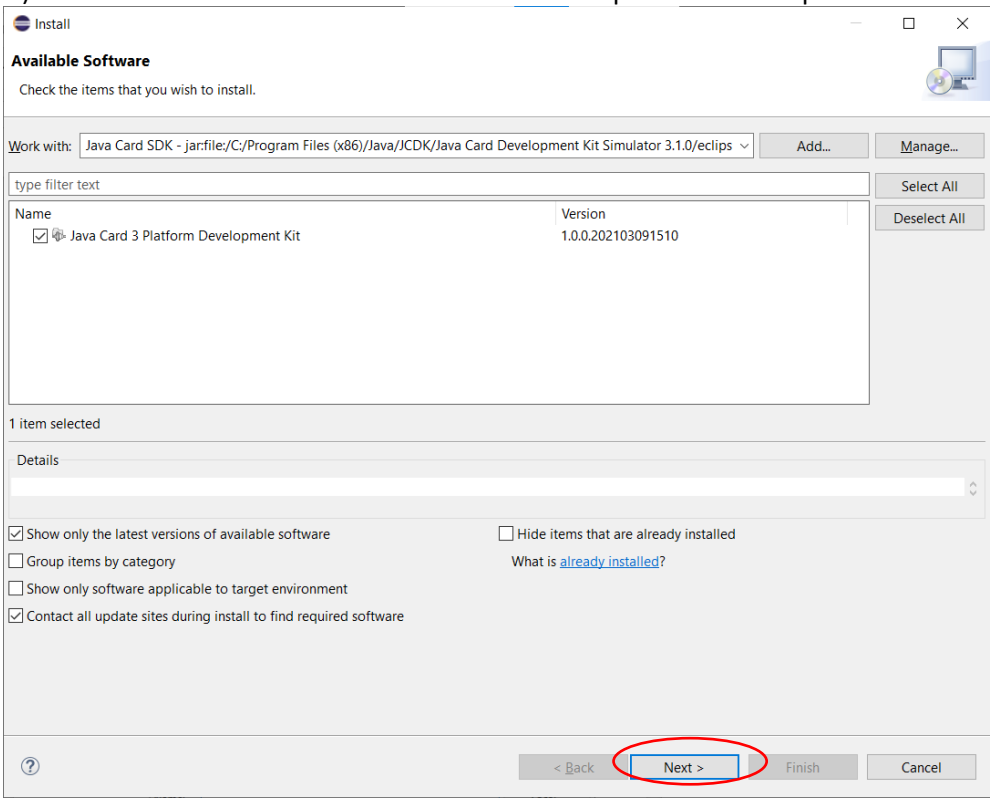
JC_HOME\eclipse-plugin\jcdk-repository_yyyymmddxxxx.zip

E) Ajouter comme Nom: `Java Card SDK`

F) Cliquer sur «Ajouter»



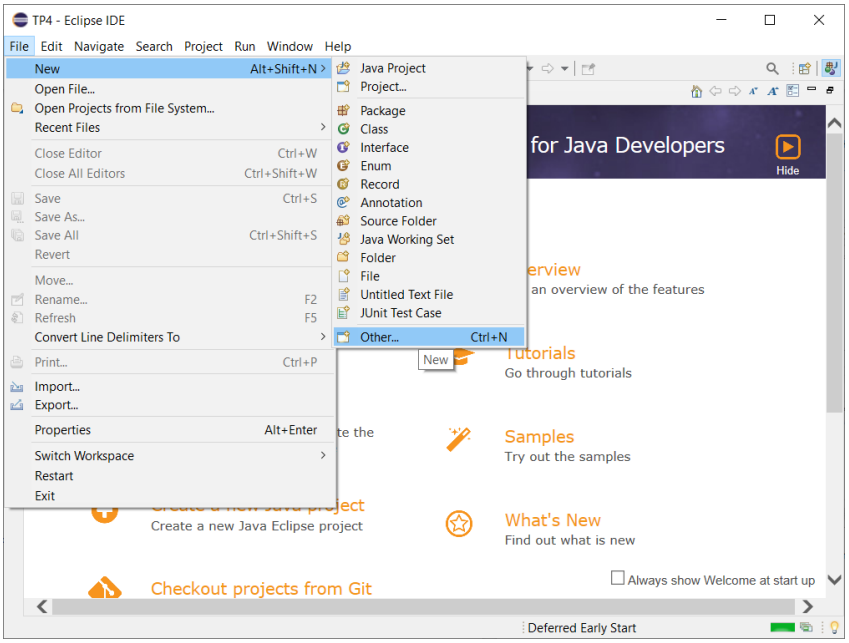
G) Selectionner Java Card 3 Platform Development Kit et cliquer sur «Suivant»



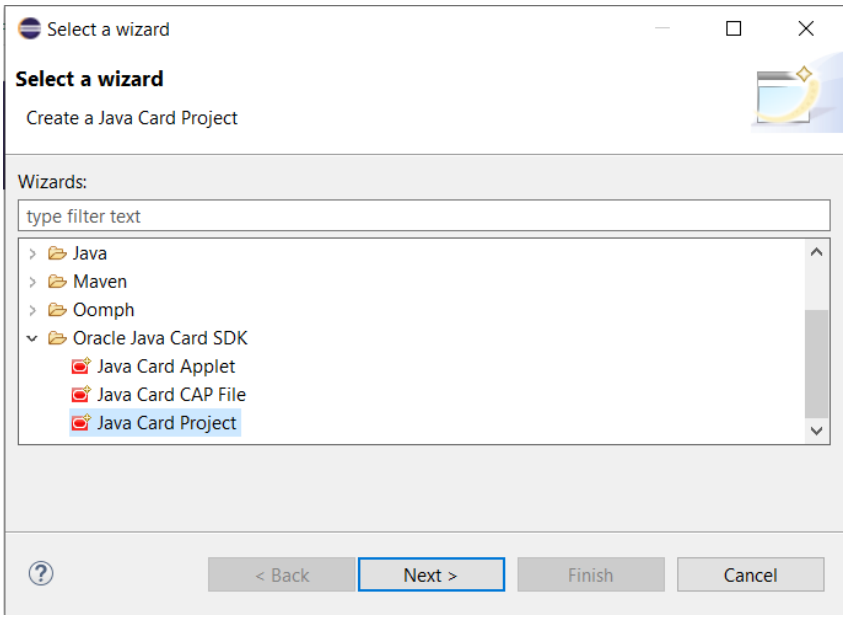
H) Attendre l'achèvement de l'installation et redémarrer Eclipse quand il vous de demande ceci

7) Création du premier projet JavaCard

A) Cliquer sur «Fichier -> Nouveau -> Autre»



B) Cliquer sur «Oracle Java Card SDK -> Java Card Applet»



C) Cliquer dur Suivant puis sur Créer

Conclusion :

A ce stade, nous disposons d'un environnement de développement complet permettant de créer des Applet Javacard, de les simuler et de créer des applications clientes.

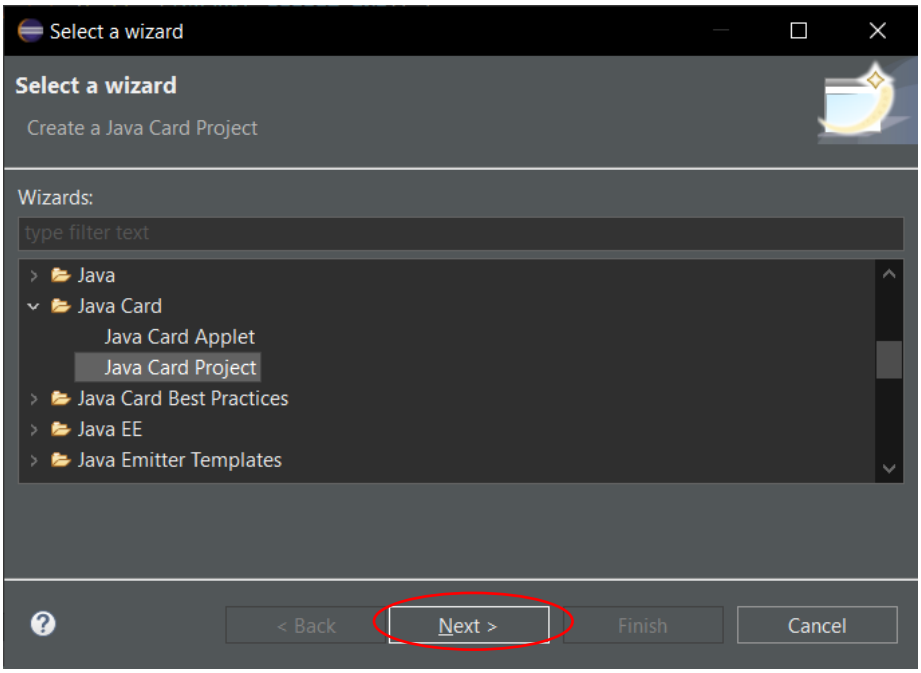
Objectifs achevés

benhamoudadhia
[Email address]

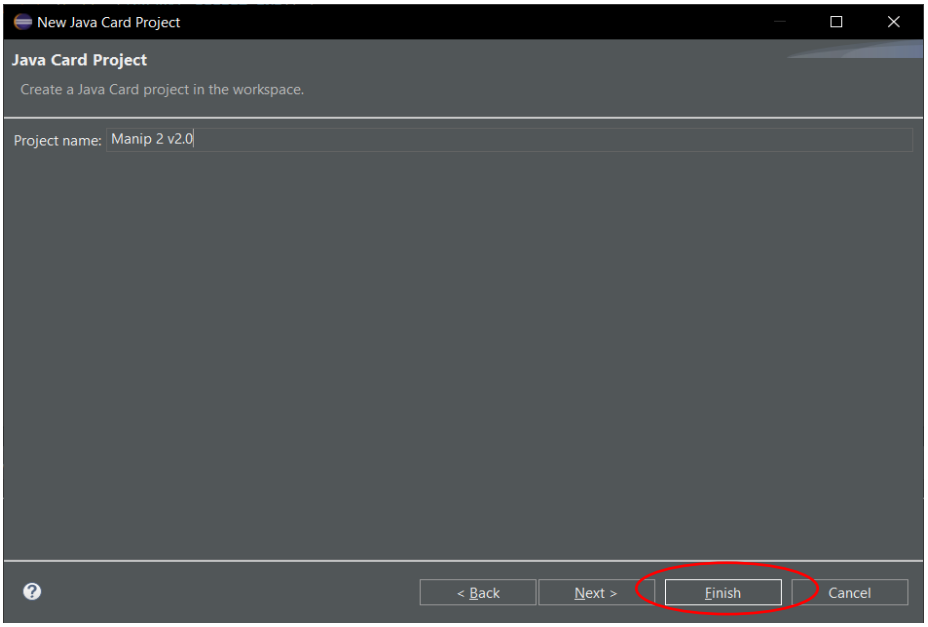
Table des Matières

Création de notre Projet JavaCard

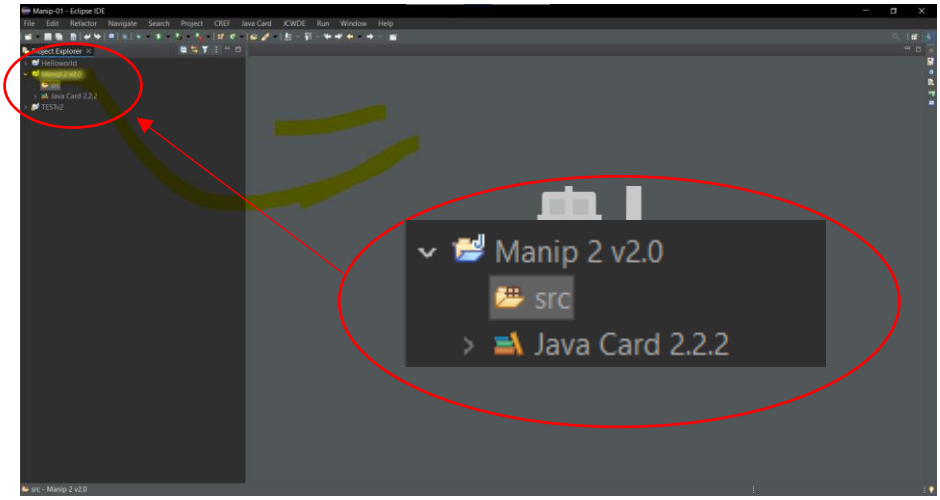
On Créer notre premier Projet JavaCard



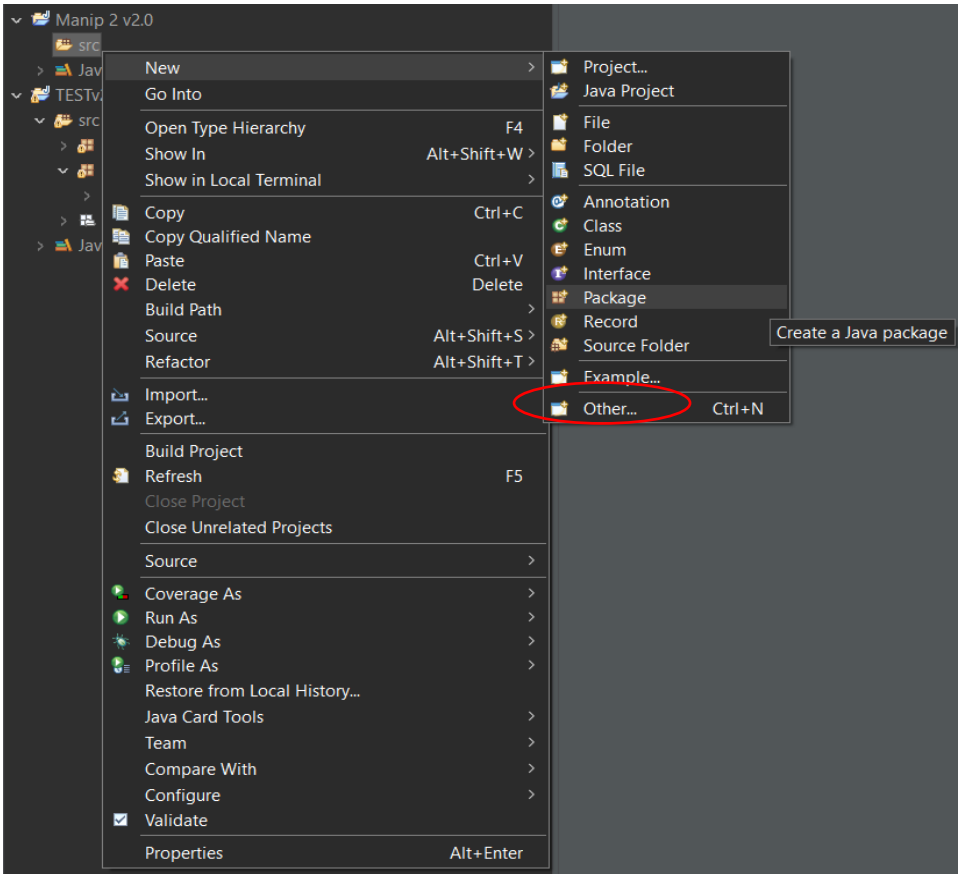
On nomme notre Premier Projet



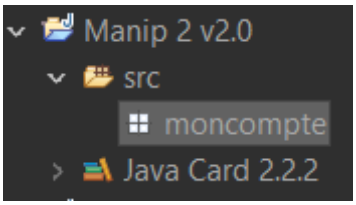
Maintenant on dispose de notre premier projet



Afin de finaliser cette étape, on créera un package où notre première JavaCard Applet se trouvera

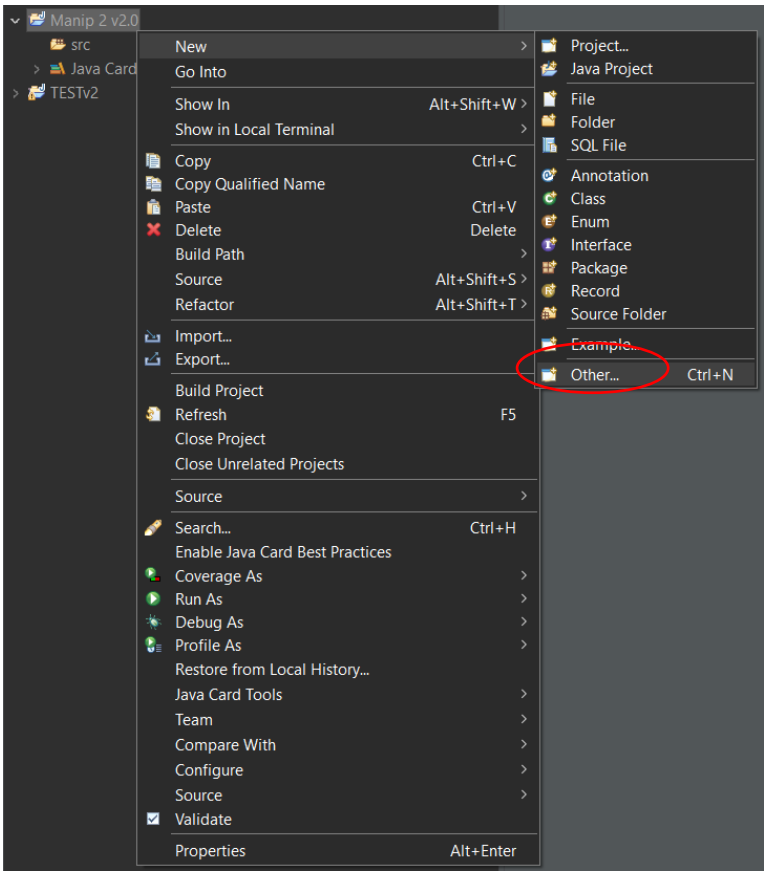


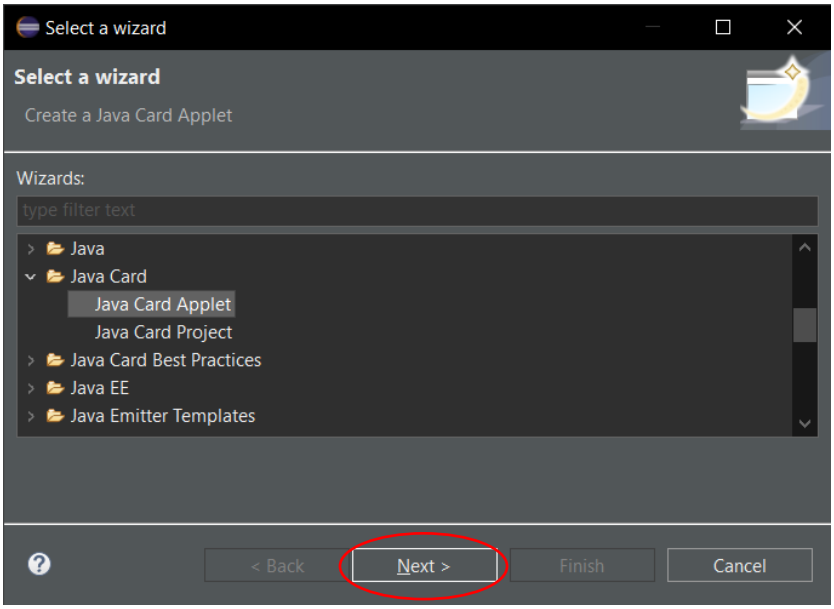
Etat Final :

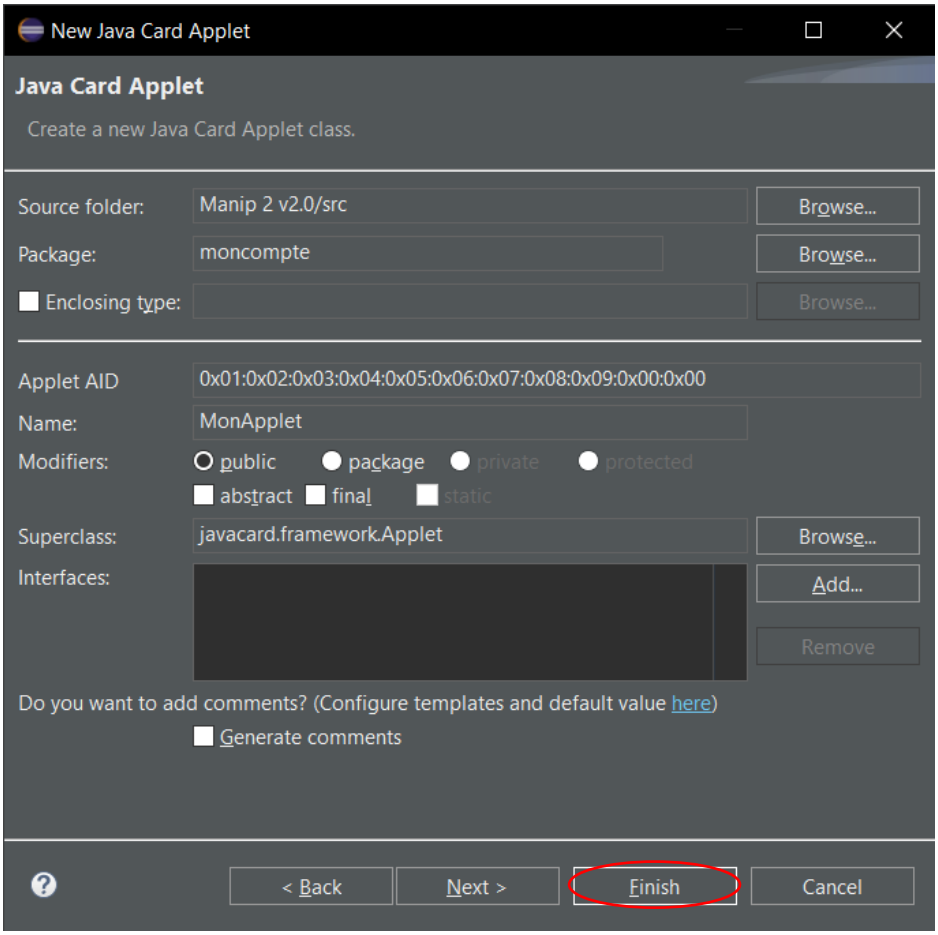


Création de notre première Applet JavaCard

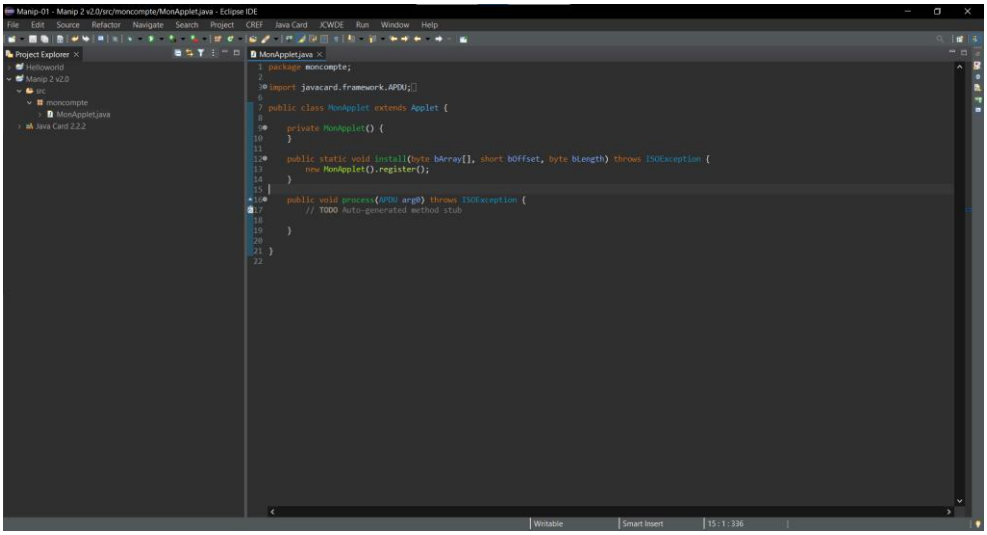
Clique droit sur notre projet -> Nouveau -> Autres…







Eclipse nous génère alors la squelette de notre Applet automatiquement



Implémentation de notre Applet

Cette Applet jouera le simple rôle d'un compteur.

Cette Applet doit alors implémenter 4 méthodes :

- Incrémenter le compteur
- Décrémenter le compteur
- Interroger le compteur
- Initialiser le compteur à une valeur donnée

A. Importer l'API JavaCard

```
import javacard.framework.APDU;
import javacard.framework.Applet;
import javacard.framework.ISOException;
```

Eclipse a déjà importé des APIs, cependant on importera une autre :

```
import javacard.framework.ISO7816;
```

Qui est une interface qui définit des constantes dont les valeurs représentent les valeurs d'exceptions courantes.

B. Déclarer les attributs et les constantes

On déclare les constantes suivantes :

- Identificateur de l'Applet : permet d'indentifier la familles des commandes APDU supportées par notre applet JavaCard

```
/* Constantes */
 public static final byte CLA_MONAPPLET = (byte) 0xB0;
```

- Identificateurs des instances des méthodes de notre Applet : permet à l'applet de reconnaitre quelle méthode on souhaite excécuter

```
public static final byte INS_INCREMENTER_COMPTEUR = 0x00;
public static final byte INS_DECREMENTER_COMPTEUR = 0x01;
public static final byte INS_INTERROGER_COMPTEUR = 0x02;
public static final byte INS_INITIALISER_COMPTEUR = 0x03;
```

- Les Attributs : les attributs utilisés dans notre applet

```
/* Attributs */
private byte compteur;
```

C. Définition des méthodes publiques qu'on doit implémenter

- Méthode *install()*

```
public static void install(byte bArray[], short bOffset, byte bLength) throws ISOException {
    new MonApplet().register();
}
```

- Méthodes *process()*

```
    public void process(APDU apdu) throws ISOException {
        // TODO Auto-generated method stub
        byte[] buffer = apdu.getBuffer();

        if (this.selectingApplet()) return;

        if (buffer[ISO7816.OFFSET_CLA] != CLA_MONAPPLET) {
            ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
        }

        switch (buffer[ISO7816.OFFSET_INS]) {
            case INS_INCREMENTER_COMPTEUR:
                compteur++;
                break;

            case INS_DECREMENTER_COMPTEUR:
                compteur--;
                break;

            case INS_INTERROGER_COMPTEUR:
                buffer[0] = compteur;
                apdu.setOutgoingAndSend((short) 0, (short) 1);
                break;
            case INS_INITIALISER_COMPTEUR:
                apdu.setIncomingAndReceive();
                compteur = buffer[ISO7816.OFFSET_CDATA];
                break;
            default:
                ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
        }
    }
}
```

Une autre implémentation de la méthode INTERROGER_COMPTEUR :

```
case INS_INITIALISER_COMPTEUR:
    apdu.setIncomingAndReceive();
    compteur = buffer[ISO7816.OFFSET_P1];
    break;
```

Cette implémentation sera utilisée pour le reste du compte Rendu

Dans cette méthode on trouve le cœur de notre Applet, ceci est le script qui tournera lors de l'appel de notre applet.

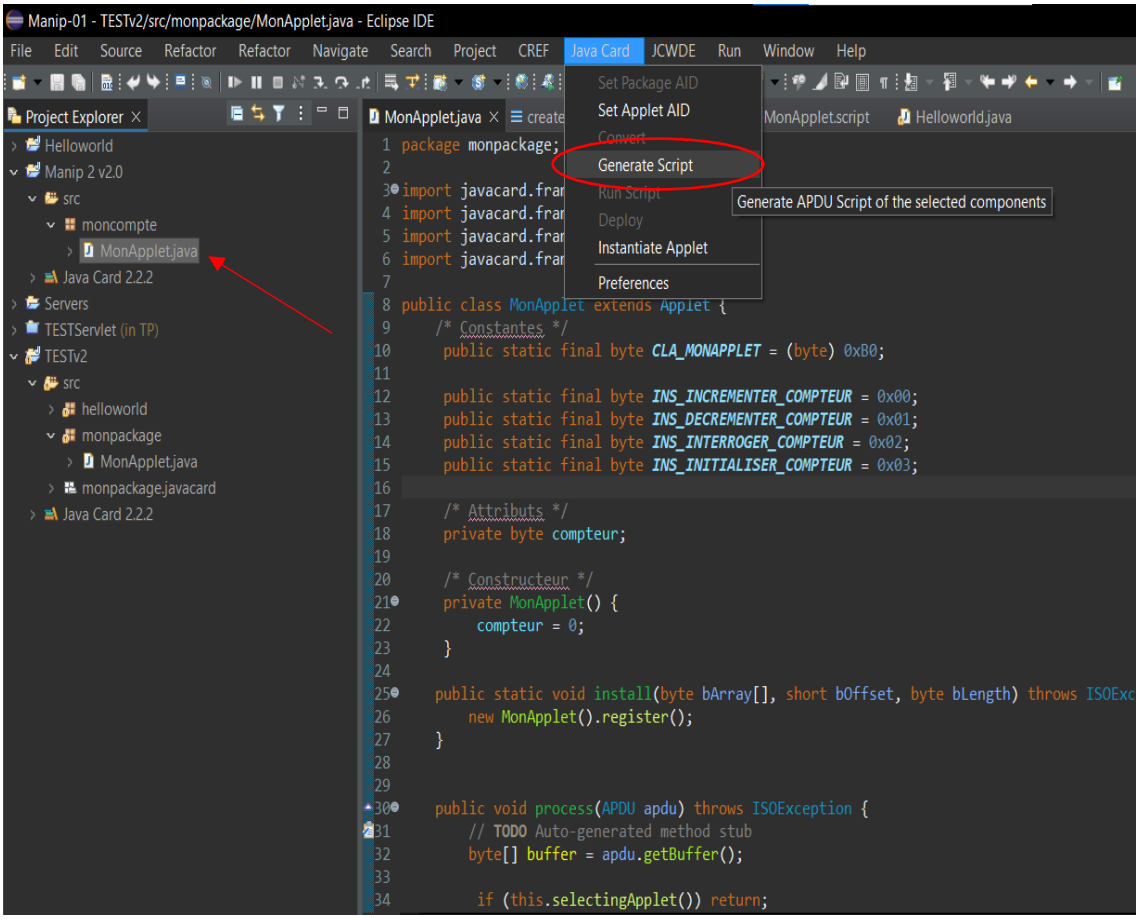
L’applet extrait le buffer qui contient les données en entrées.
Ensuite il se renseigne sur l’identifiant de la méthode sélectionner et l’exécute.(Le switch case) tout en gérant les exceptions.

D. Outils de simulation

Avant de pouvoir tester notre implémentation, on doit générer les APDU responsables sur la création d »’une instance de notre Applet et puis sa sélection de notre Applet qui est installée sur notre carte JavaCard virtuelle.

Pour faire ceci, on appui sur :

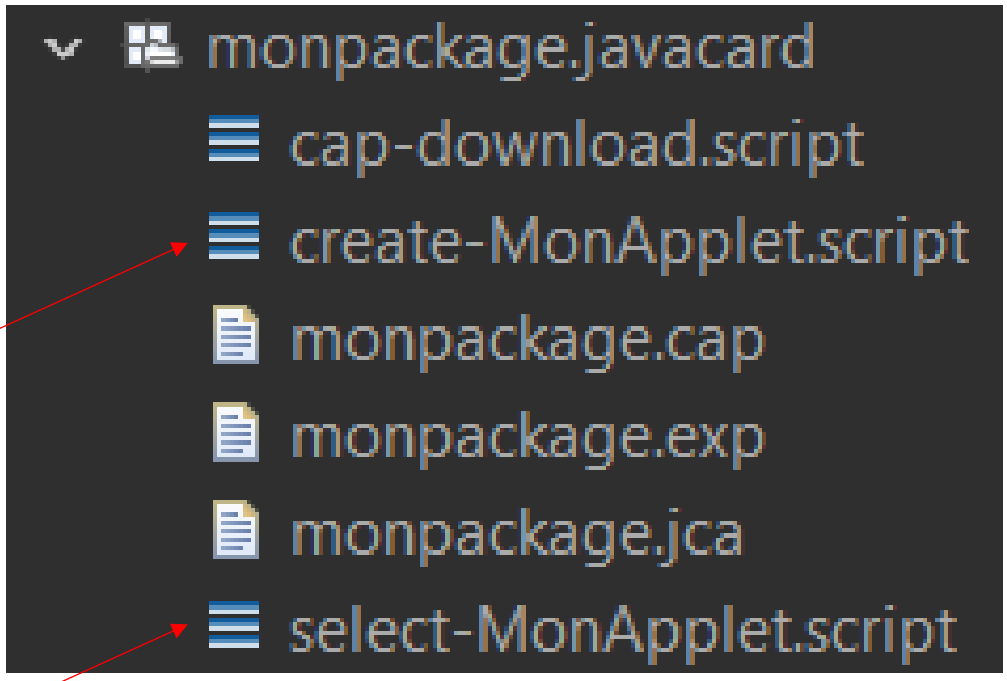
Notre Applet -> Menu JavaCard -> Générer Script



Alors, 3 scripts seront générés sous un nouveau package nommée : monpackage.javacard :

- cap-download.script : upload de l’applet
- create-MonApplet.script : instanciation (installation) de l’applet
- select-MonApplet.script : sélection de l’applet

Pour le moment on s’intéressera pour les deux derniers puisque le JCWDE s’occupe lui-même de l’upload de l’applet:



1. *Sans conservation d'état*

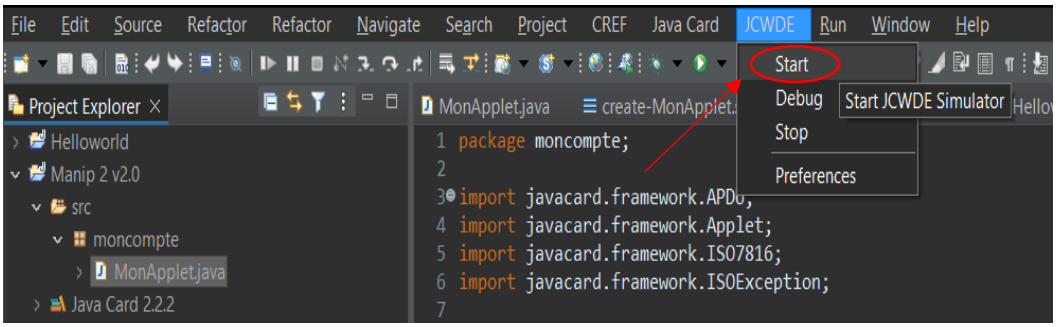
On doit prendre en compte ces outils de simulation :

- *APDUTOOL :* Il gère l'échange des APDUs entre l'applet et la carte JavaCard (virtuelle ou réelle)
- *JCWDE (Java Card Workstation Development Environment) :* Programme qui simule le JCRE dans une JVM
- *CREF (Java Card Platform Simulator):*

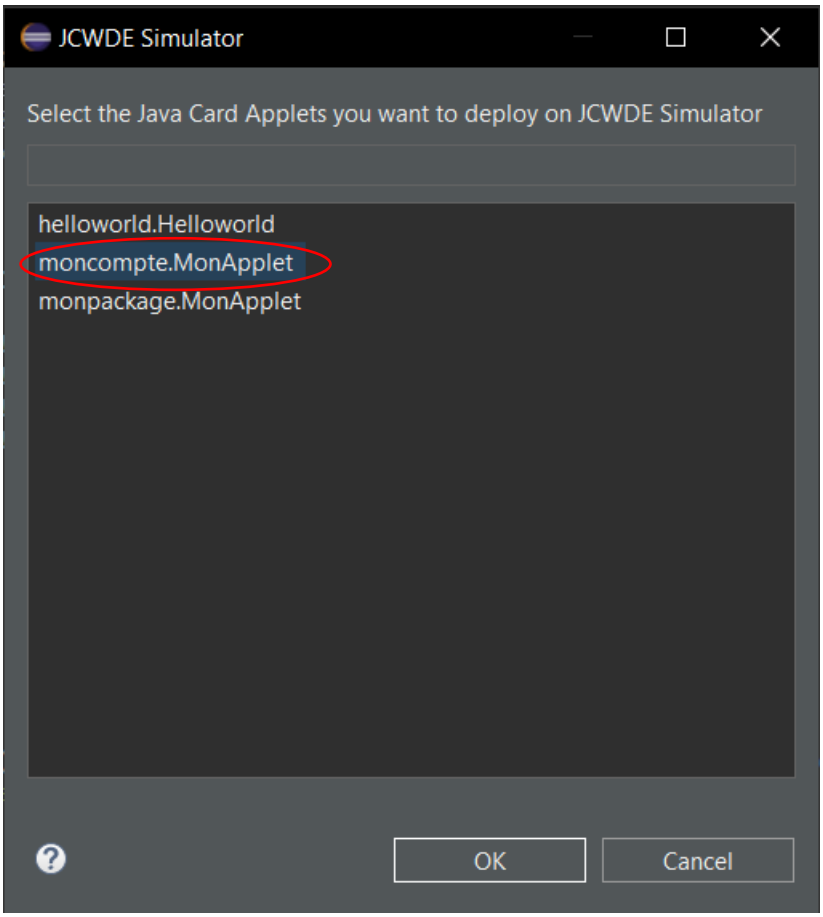
Concept

Ces 3 étapes sont à suivre :

Lancer le JCWDE :



Sélectionner son Applet et appuyer sur ok pour lancer le simulateur :

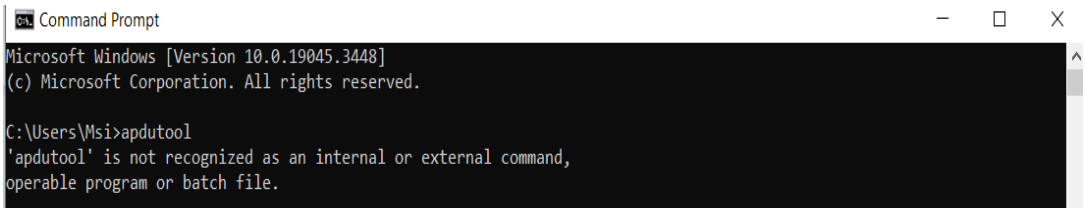


- Lancer une simulation de connexion (APDUTOOL) :
  Ouvrir le CMD et lancer la commande apdutool

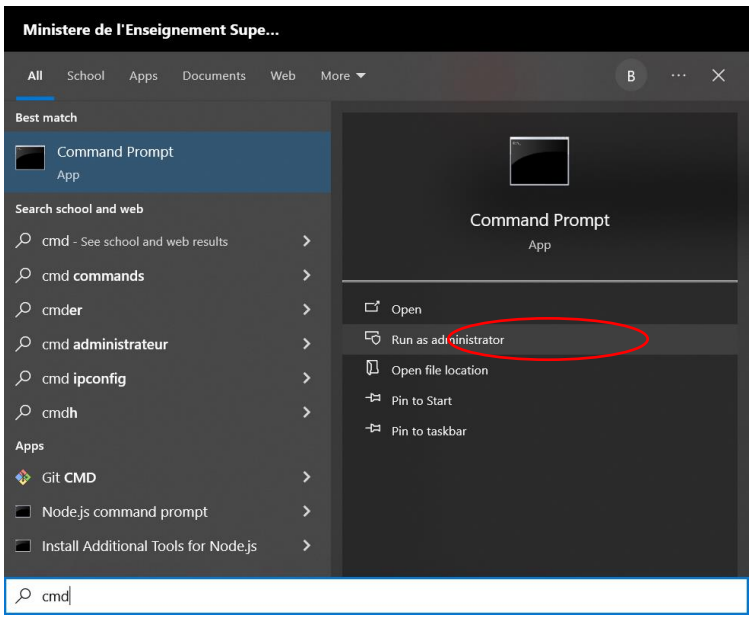
```
C:\Users\Msi>apdutool
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
Opening connection to localhost on port 9025.
java.net.ConnectException: Connection refused: connect
```

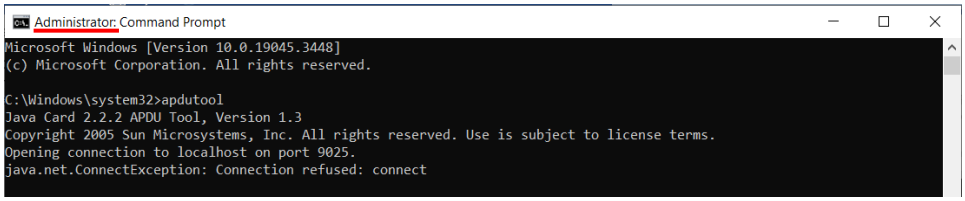
- Si vous rencontrez un message d'erreur comme ci-dessous et que vous êtes sur d'avoir bien installer JavaCard sur votre machine, alors essayer de lancer l'invite de commande (CMD) en tant qu'Adminastrateur

Erreur :



Solution :





- *Installer et sélectionner notre Applet :*

  Copier et coller les contenus des deux fichiers (sauf la commande « powerdown ; »)
  - Create-MonApplet.script
  - Select-MonApplet.script

create-MonApplet.script

```
powerup;
// Select the installer applet
0x00 0xA4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
// create MonApplet applet
0x80 0xB8 0x00 0x00 0xd 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x00 0x7F;
```

select-MonApplet.script

Aussi taper la commande « powerup » une seul fois

```
powerup;
// select MonApplet applet
0x00 0xA4 0x04 0x00 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x7F;
```

- Pour chaque APDU envoyé, un message est retourné.
- Le code de ce message est spécifié dans les champs SW1 et SW2
- Ce code message est en Hexadecimal
- On concatine les deux champs pour avoir le code message
- Le code de réuissite est --> 0x9000

Résultat attendu :

```
C:\Users\Msi>apdutool
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
Opening connection to localhost on port 9025.
Connected.
powerup;
// Select the installer applet
0x00 0xA4 0x04 0x00 0x09 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
// create MonApplet applet
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le: 00, SW1: 90, SW2: 00
0x80 0xB8 0x00 0x00 0xd 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x00 0x7F;
CLA: 80, INS: b8, P1: 00, P2: 00, Lc: 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 02, 00, Le: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 02, SW1: 90, SW2: 00
// select MonApplet applet
0x00 0xA4 0x04 0x00 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x7F;
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 02, Le: 00, SW1: 90, SW2: 00
```

```
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 09, a0, 00, 00, 00, 62, 03, 01, 08, 01, Le: 00, SW1: 90, SW2: 00
```

```
SW1: 90, SW2: 00
```

Code de réussite !

Maintenant notre applet est installée et déployée.

On peut commencer à tester notre Applet.

- Structurer nos APDUs

Command APDU						
Mandatory header				Optional body		
CLA	INS	P1	P2	Lc	Data field	Le

CLA : Classe d'instruction (Définie dans notre constante CLA_MONAPPLET)

```
CLA_MONAPPLET = (byte) 0xB0;
```

INS : Code de l'instruction (Définie dans notre constante INS_*)

```
INS_INCREMENTER_COMPTEUR = 0x00;
INS_DECREMENTER_COMPTEUR = 0x01;
INS_INTERROGER_COMPTEUR = 0x02;
INS_INITIALISER_COMPTEUR = 0x03;
```

P1, P2 : Paramètre 1 et 2 respectivement

Voici un exemple d'un APDU

0xB0 0x02 0x00 0x00 0x00 0x7F;

- Envoyer nos APDUs

Afin d'envoyer nos APDUs, il suffit de saisir notre APDU dans l'invite de commande où on a installer et sélectionner notre Applet et appuyer sur entrer après.

TEST

Pour tester notre applet on va lancer les commandes suivantess :

- Interroger le compteur
- Incrémenter le compteur
- Interroger le compteur
- Décrémenter le compteur
- Interroger le compteur
- Initialiser le compteur à 50
- Interroger le compteur

Voici les commandes à saisir

- 0xB0 0x02 0x00 0x00 0x00 0x7F;
- 0xB0 0x00 0x00 0x00 0x00 0x7F;
- 0xB0 0x02 0x00 0x00 0x00 0x7F;
- 0xB0 0x01 0x00 0x00 0x00 0x7F;
- 0xB0 0x02 0x00 0x00 0x00 0x7F;
- 0xB0 0x03 0x32 0x00 0x00 0x7F; (décimal)50 = 0x32
- 0xB0 0x02 0x00 0x00 0x00 0x7F;

Résultat :

```
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 00, SW1: 90, SW2: 00
```

```
0xB0 0x00 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 00, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
```

Compteur incrémenté

```
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 01, SW1: 90, SW2: 00
```

```
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 32, SW1: 90, SW2: 00
```

Compteur incrémenté

```
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 00, SW1: 90, SW2: 00
```

```
0xB0 0x03 0x32 0x00 0x00 0x7F;
CLA: b0, INS: 03, P1: 32, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
```

Compteur initialisé à 50

2. *Avec Conservation d'état*

Pour ceci on va utiliser l'outil CREF.

Concept :

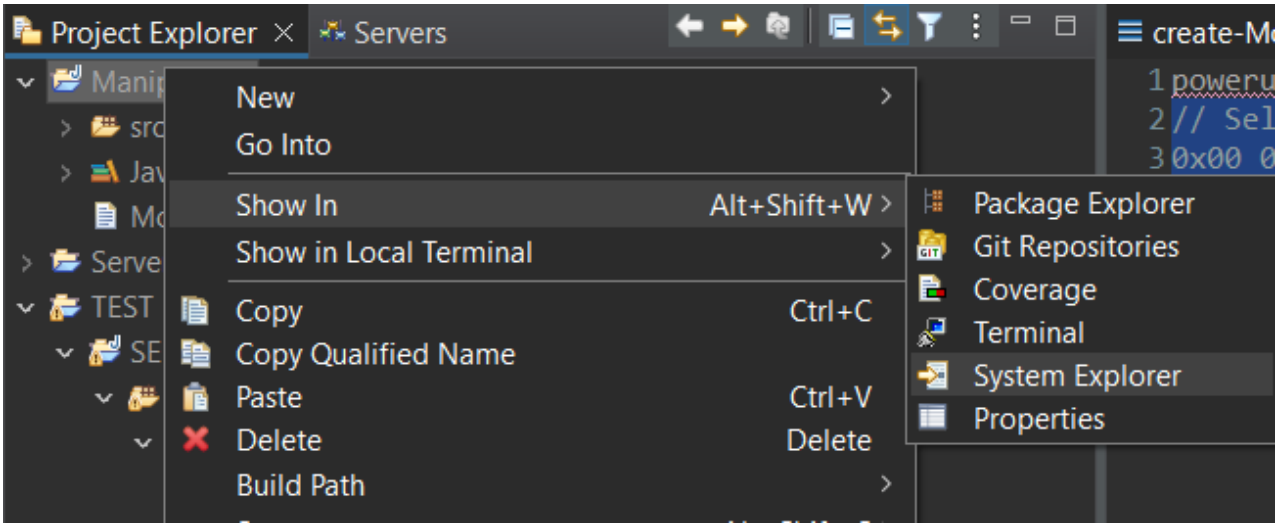
Pour Ceci on suivra les étapes suivantes :

- Générer un fichier eeprom et enregistrer l'application dessus
- Simuler la carte en chargant le fichier eerpom généré

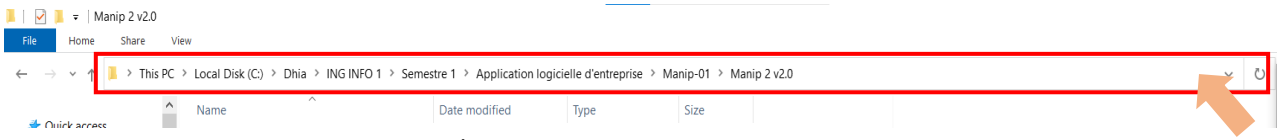
TEST :

Tout d'abbord on créer notre fichier eeprom :

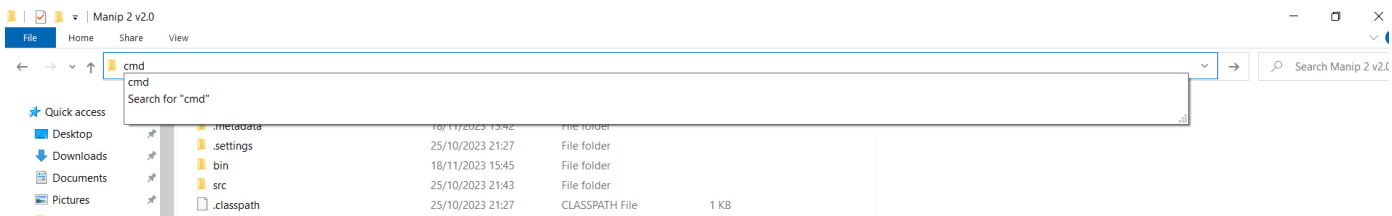
- On ouvre le dossier de notre projet JavaCard



- On clique sur la barre de chemin

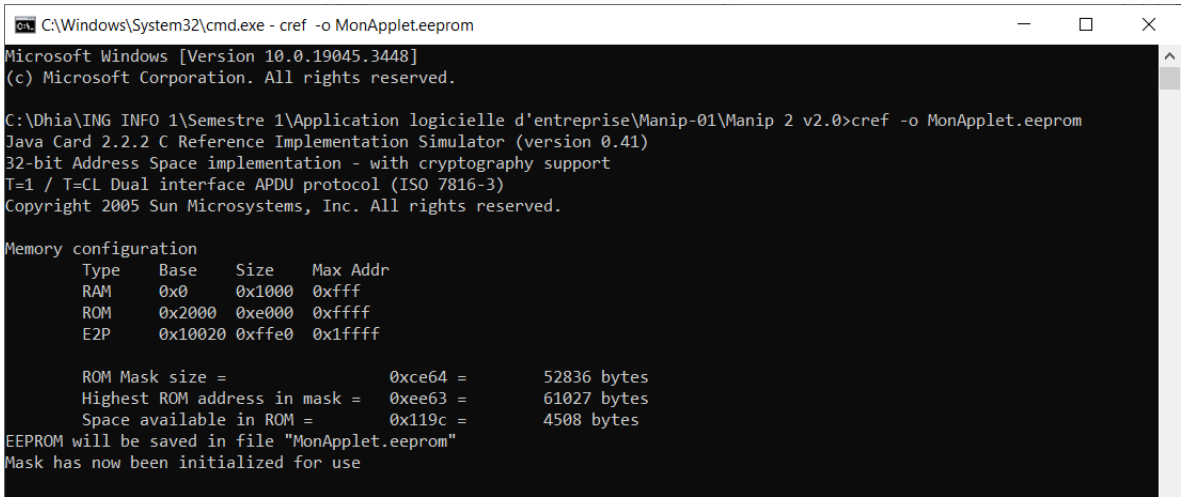


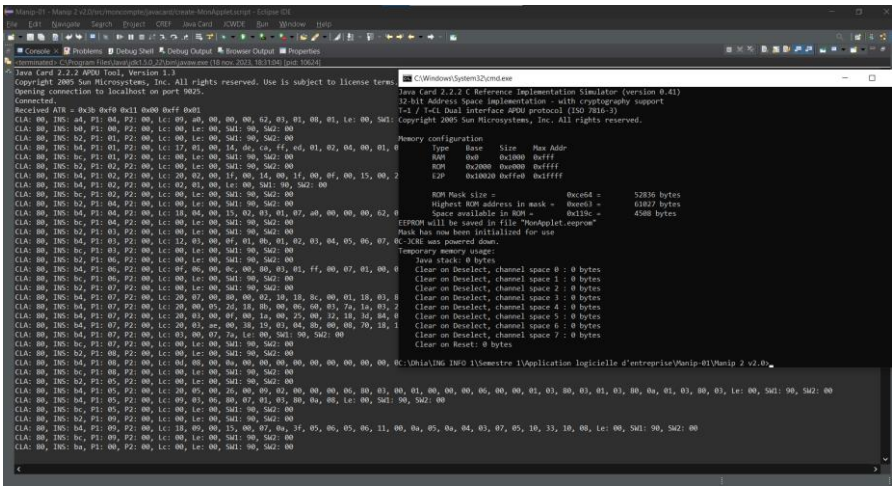
- On saisie « cmd » et on appuie sur entrer



Maintenant, on dispose d’une invite de commande situé sous le dossier de notre projet

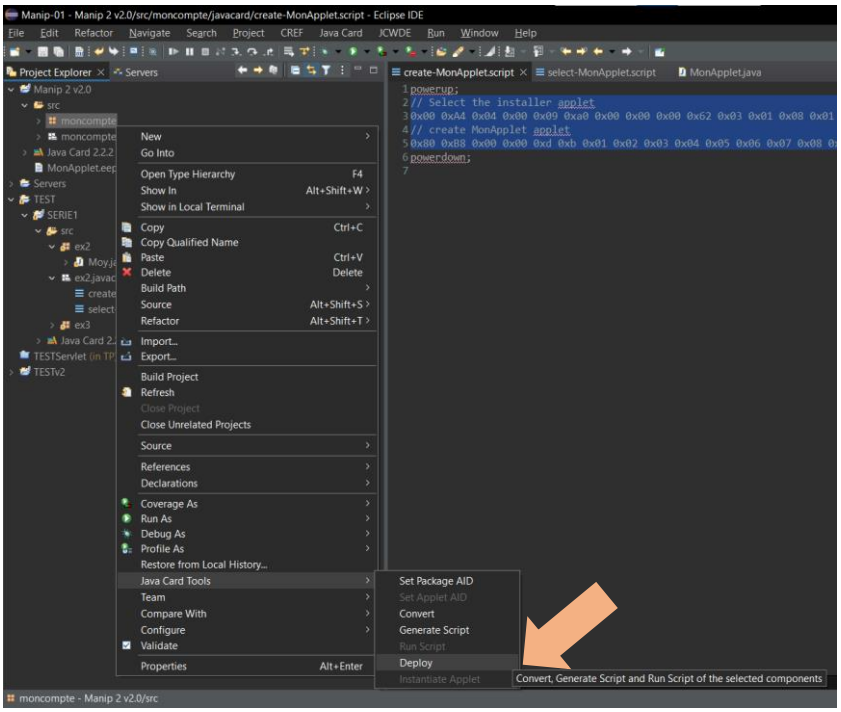
On Créer notre fichier « eeprom » avec la commande CREF -o :





Notre fichier est en attente que notre Applet soit déployé .

- **De retour en Eclipse**, on clique droit sur notre Package et on appuie sur deployer sous le menu Java Card Tools



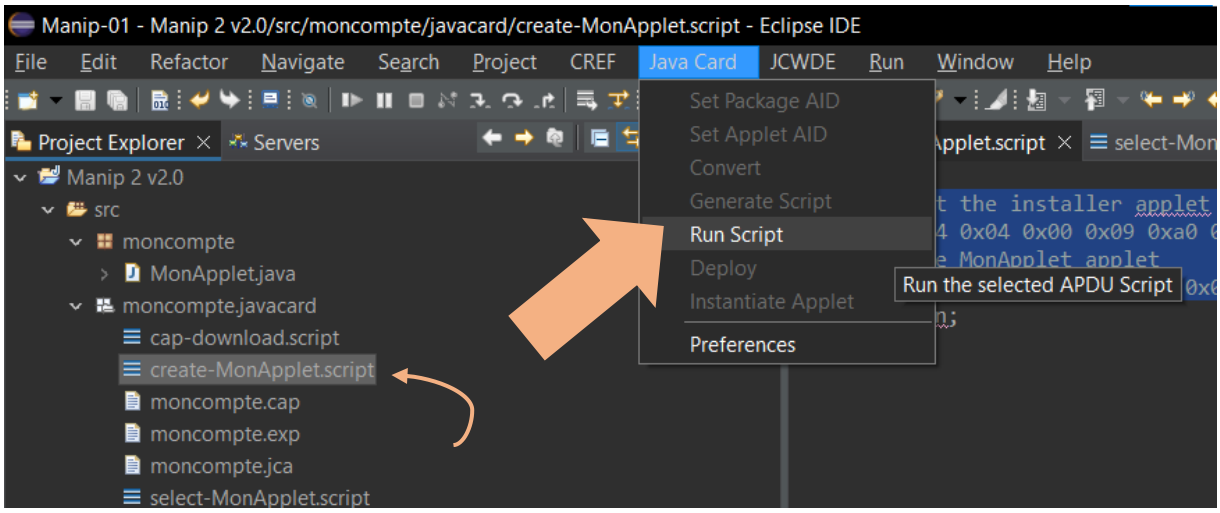
Ceci sont les message de retour des APDUs d'upload

- **On relance CREF** mais avec l'option -i avec qu'il se relance à partie du fichier eeprom qu'on vient juste de créer

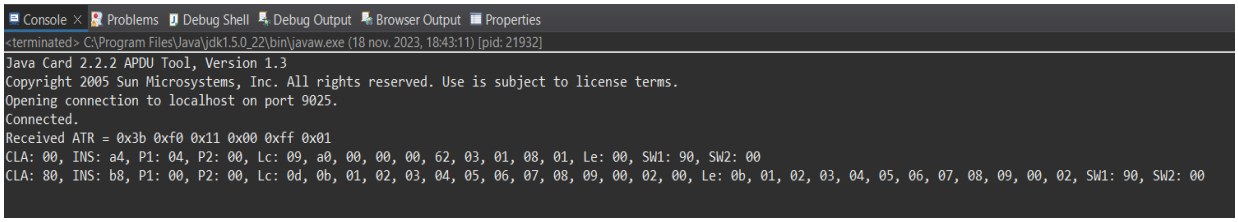
```
C:\Dhia\ING INFO 1\Semestre 1\Application logicielle d'entreprise\Manip-01\Manip 2 v2.0>cref -i MonApplet.eeprom
Java Card 2.2.2 C Reference Implementation Simulator (version 0.41)
32-bit Address Space implementation - with cryptography support
T=1 / T=CL Dual interface APDU protocol (ISO 7816-3)
Copyright 2005 Sun Microsystems, Inc. All rights reserved.

Memory configuration
       Type    Base    Size    Max Addr
       RAM     0x0     0x1000  0xfff
       ROM     0x2000  0xe000  0xffff
       E2P     0x10020 0xffe0  0x1ffff

       ROM Mask size =                0xce64 =        52836 bytes
       Highest ROM address in mask =  0xee63 =        61027 bytes
       Space available in ROM =       0x119c =        4508 bytes
EEPROM (0xffe0 bytes) restored from file "MonApplet.eeprom"
Using a pre-initialized Mask
```

On s'aperçoit qu'il est écrit à la fin : « Using a pre-initialized Mask ». Ceci signifie que la lecture du fichier eeprom s'est bien effecuée.

- **Installons notre applet.** Pour cela, dans Eclipse, clic droit sur le script create-MonApplet.script, Java Card Tools, Run Script :

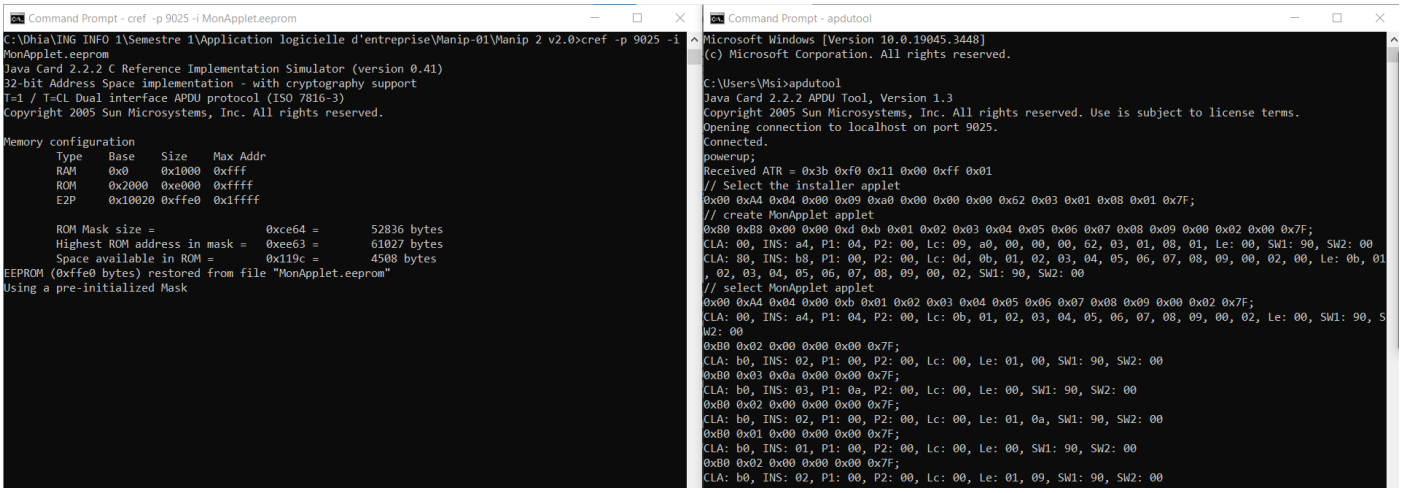


Ceci sont les message de retour des APDUs d'installation



A ce stade, l'applet est bien installée. Nous pouvons désormais commencer à tester avec l'outil « APDUTOOL »

- **On relance CREF,** avec l'option -i bien évidament
- **On lance APDUTOOL** dans une autre invite de commande
- **On lance les commandes de selection d'Applet**



Le message de retour 90 00 indique le bon déroulement des étapes

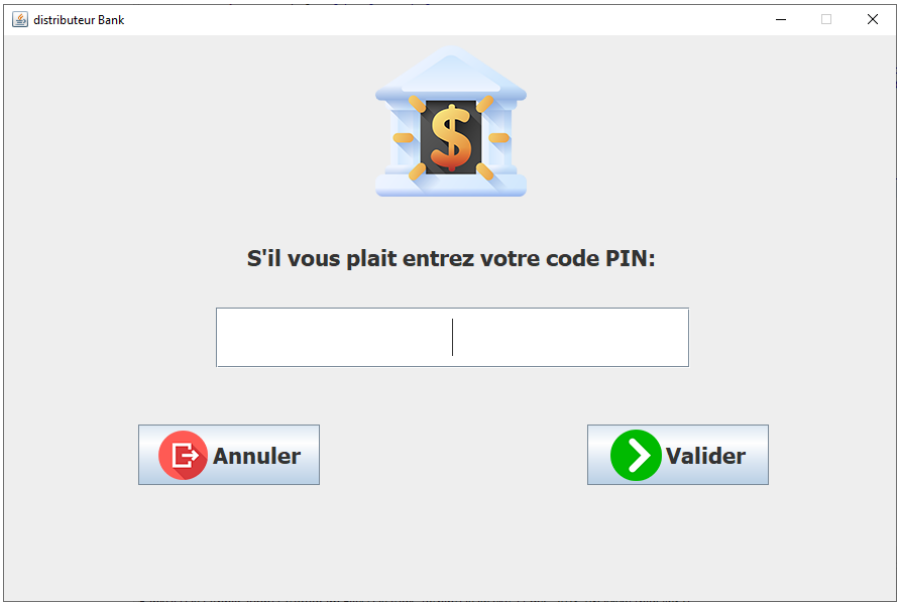
Interface Graphique

3. Vérification du Code PIN

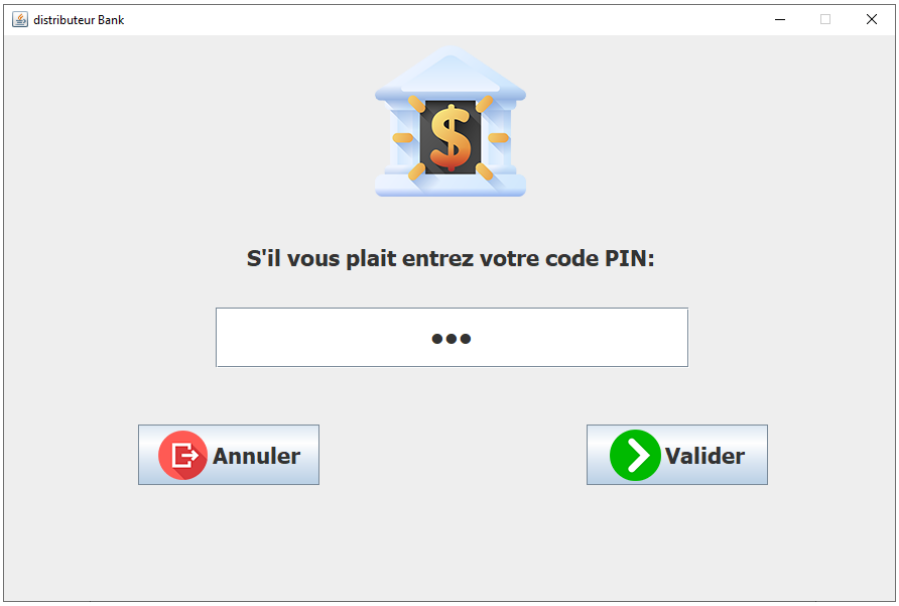
L'interface de vérification du Code PIN est la première interface qui s'affiche lors de l'exécution.

Il faut valider le code PIN afin de continuer.

Le lecteur n'accepte que les chiffres numériques même si le lecteur ne possède qu'un pavé numérique. Ceci est pour des raisons de sécurité (résoudre la vulnérabilité contre l'une injection de requête)



Le lecteur valide automatiquement le code PIN inséré à l'insertion du 4ème chiffre. (Sans appuyer



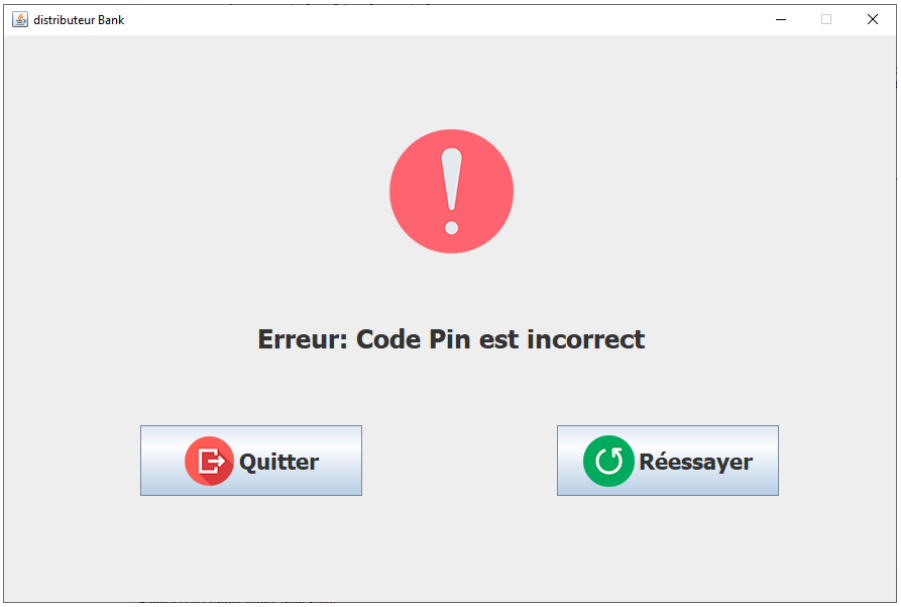
sur le bouton Valider)

- **3 Cas se présentent :**
    - ***a) Le code PIN est Correct :***

La vérification est alors terminée avec sucées et vous êtes présenté avec le menu principale du distributeur de votre banque.

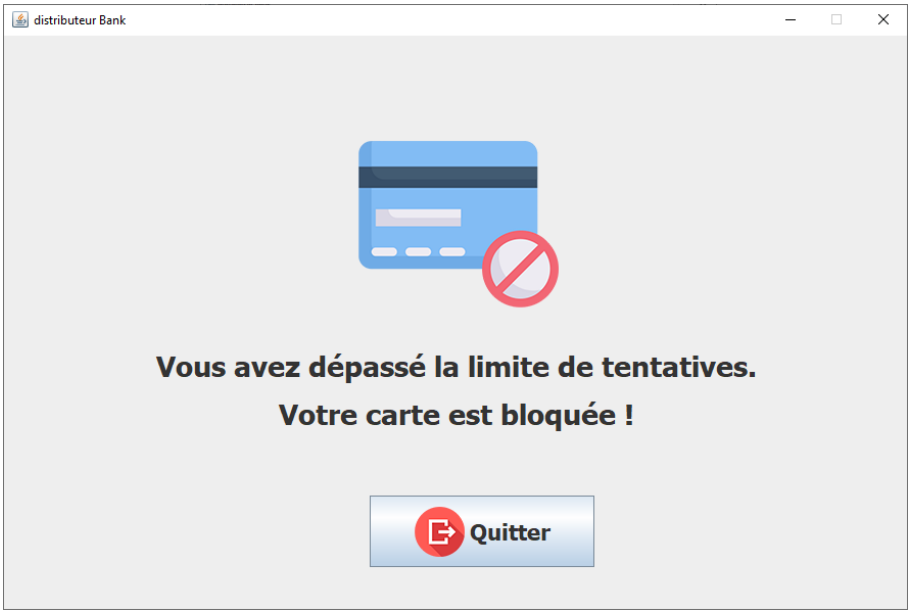
b) Le Code PIN est erroné :

Vous êtes présenté avec un message d'erreur. Aussi le nombre d'essais permis est décrémenté par 1.



c) *Le nombre maximal de tentatives est atteint :*

La carte est alors bloquée. De nouvelle tentatives de validation ne sont pas permis. Vous êtes obligé de Quitter



Toutes les interfaces suivantes (à part du menu principal) contiennent deux boutons

- **Bouton Retourner :** Permet de retourner au menu principal

- **Bouton Quitter :** Ferme l'application est envoi la commande « powerdown » à la carte

Powerdown de la carte :

```
C:\Dhia\ING INFO 1\Semestre 1\Application logicielle d'entreprise\Projet\DhiaAndElKamelCo\bin>jcwde Bank.app
Java Card 2.2.2 Workstation Development Environment, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.
jcwde is listening for T=1 Apdu's on TCP/IP port 9?025.
jcwde exiting on receipt of power down command.
```

4. Menu Solde

Le Menu Solde affiche le solde de votre compte.

- Pour afficher le solde, d**eux styles différents** peuvent se présenter :
-

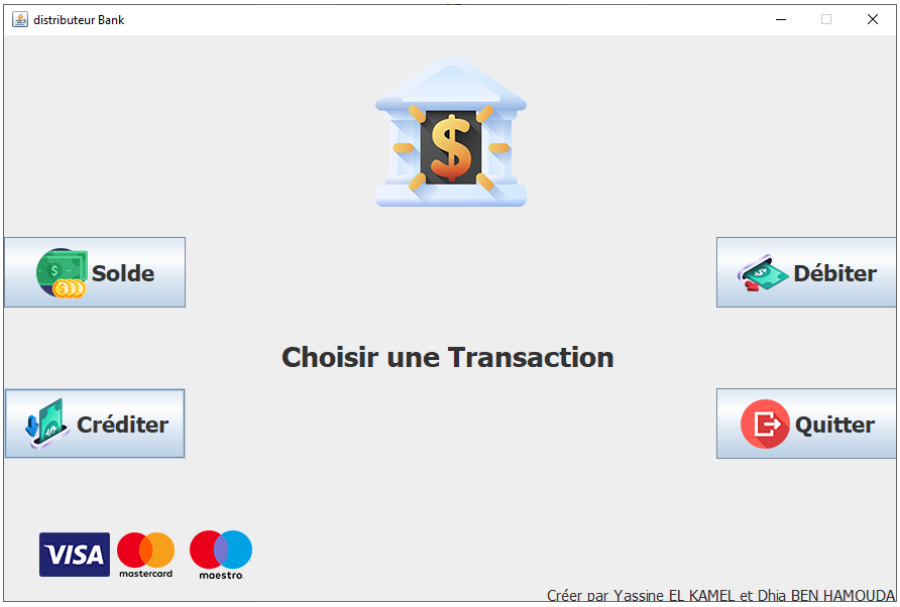
a) <u>Solde de valeur supérieur ou égal à 0 :</u>

Le solde est affiché avec un style par défaut

b) <u>Solde de valeur inférieur à 0 :</u>

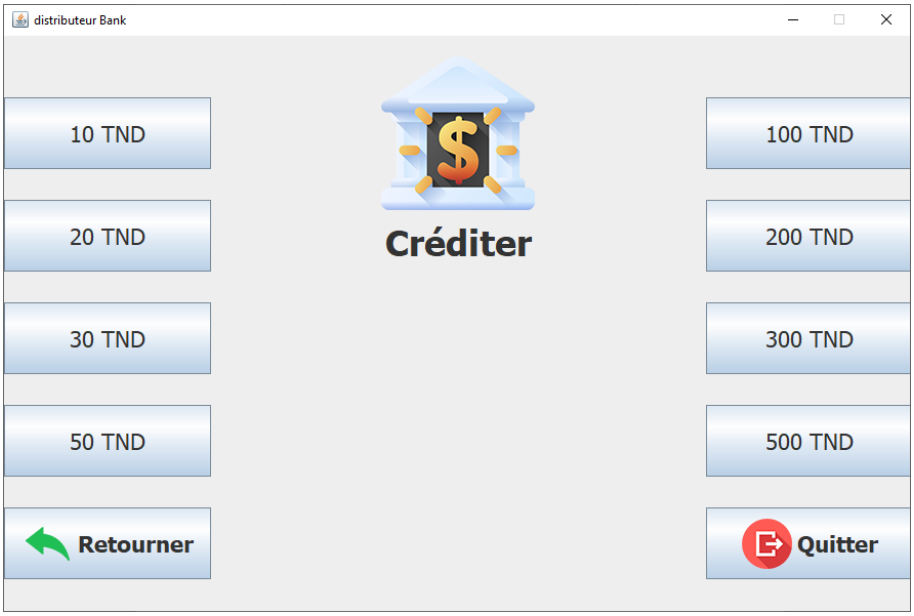
Le solde est affiché avec une couleur rouge. Le compte est dans le rouge.

5. Menu Créditer

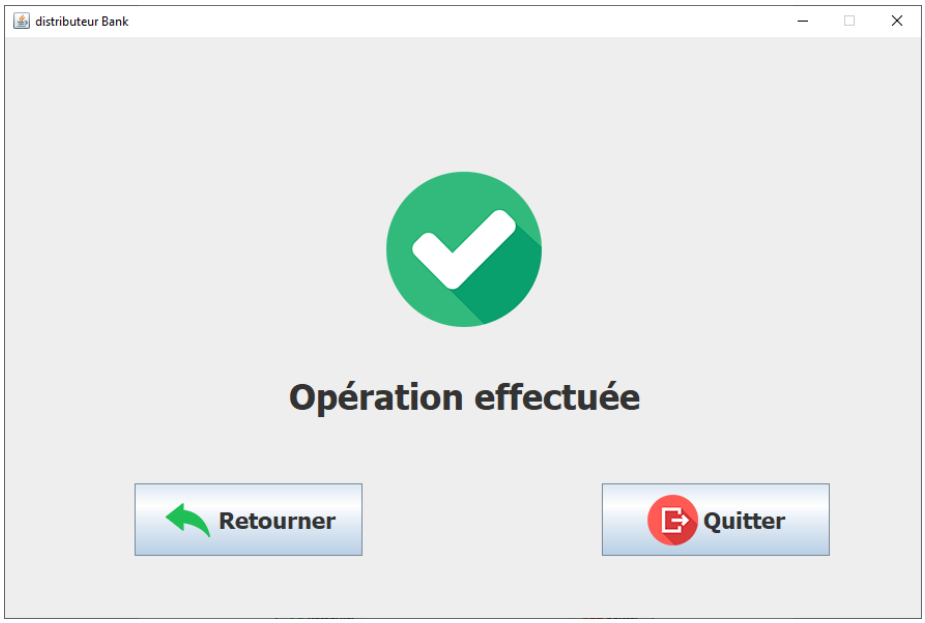


Ce menu sert à incrémenter le solde. Ceci implémente la fonctionnalité d'insérer de l'espèce dans le distributeur afin d'incrémenter le solde.

Ceci est le menu créditer

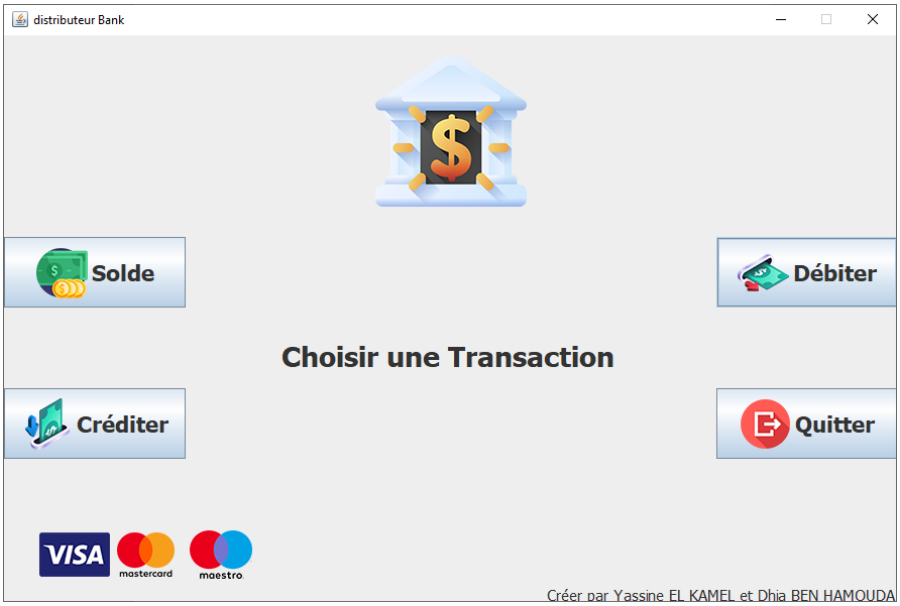


Après avoir sélectionner le montant désiré, la somme est créditée au compte et un message est affiché

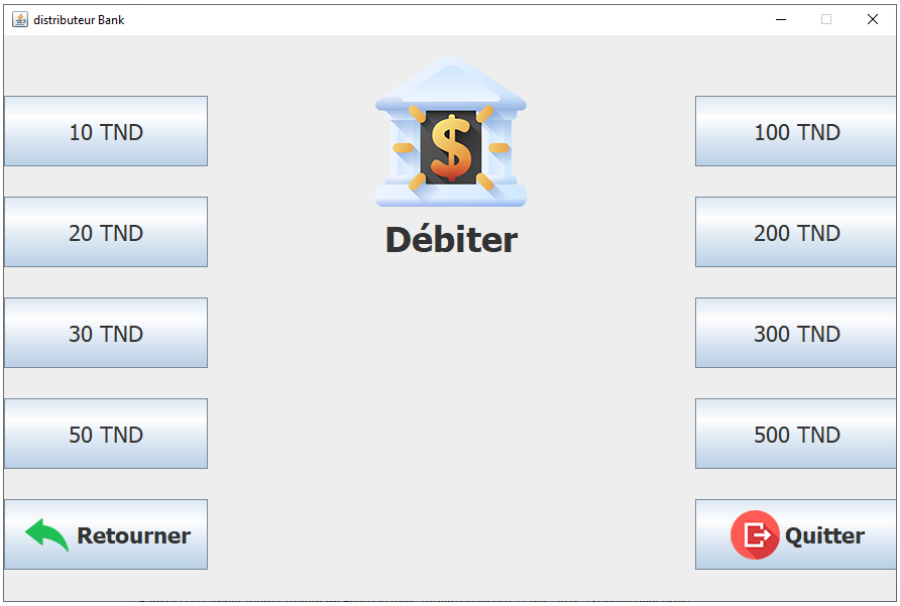


6. Menu Débiter

On retrait de l'argent à partir de ce menu.



Cliquer sur le montant désirée décrémente notre solde par ce montant.



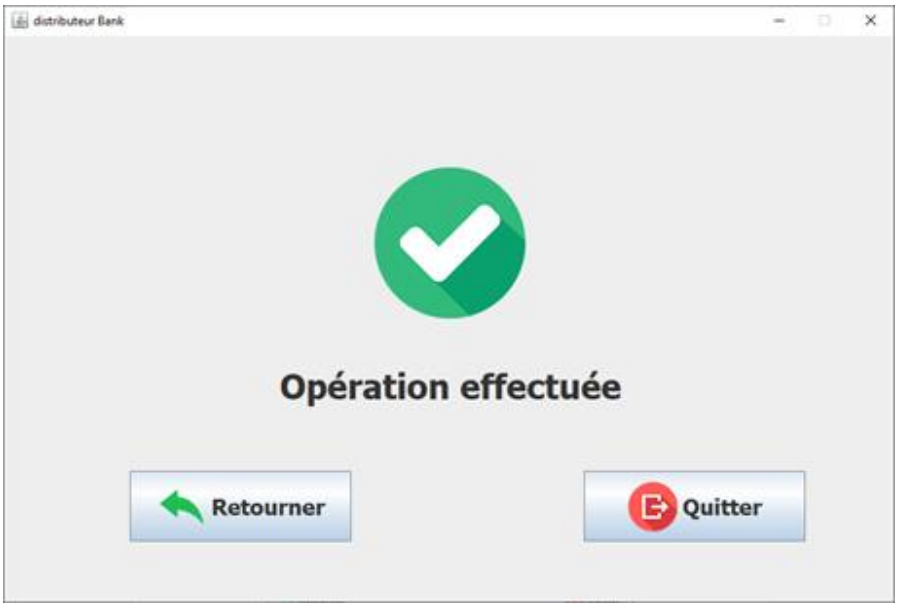
***3 Cas** se présentent :*

a) <u>Opération avec succès :</u>

Votre solde est supérieur au montant choisi.

Vous avez assez de solde à débiter.

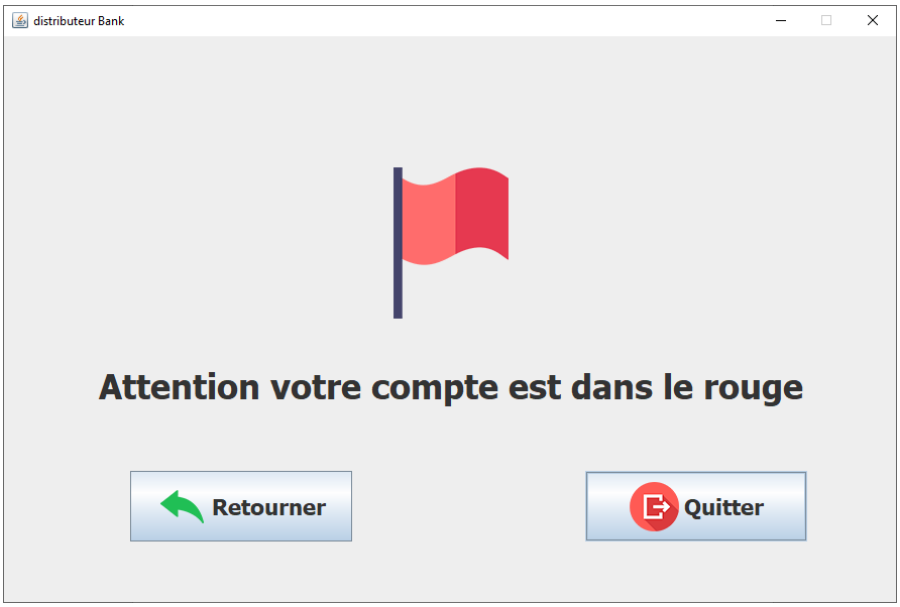
L'opération se déroule normalement.



b) <u>Votre compte est dans le rouge :</u>

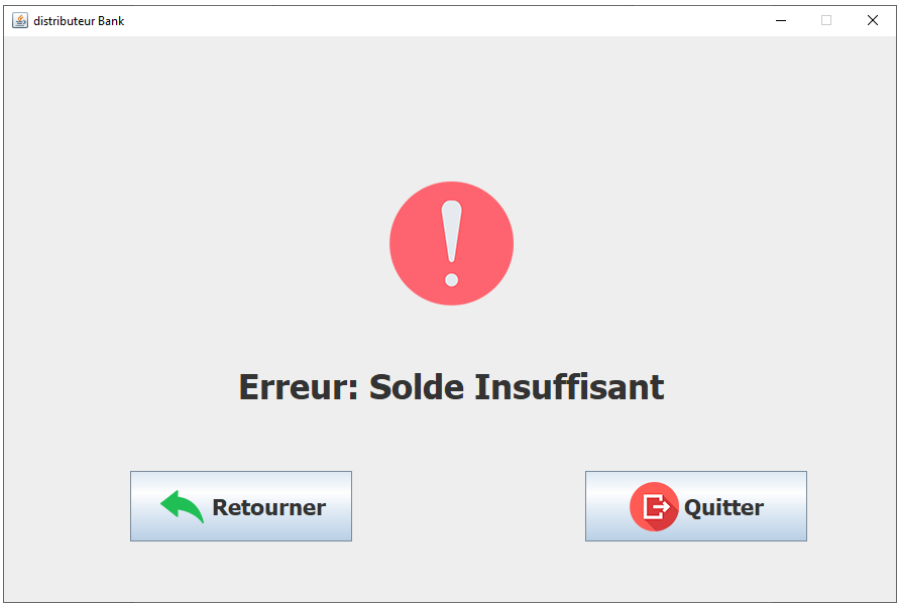
Votre solde est insuffisant, mais votre retrait d'argent ne dépasse pas le plafond de rouge permis.

L'argent est débiter accompagné d'un message d'alerte



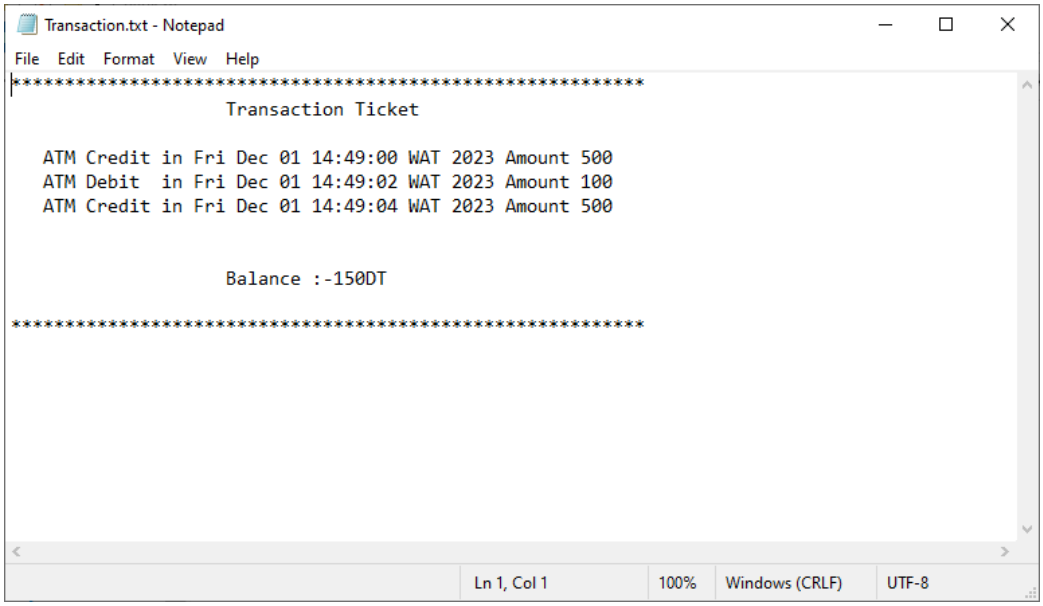
c) Solde Insuffisant :

Votre est insuffisant et dépasse le plafond de rouge permis. L'argent n'est pas débité. Un message d'erreur s'affiche.



7. Menu Imprimer mes Transactions

Ce menu permet d'imprimer l'historique des transactions, de les enregistrer dans un fichier texte et de les imprimer.



Le fichier est intitulé « Transaction.txt ». SI vous aucune imprimante n'est détectée, fichier de format « PDF » est généré.

```
************************************************************
                    Transaction Ticket

   ATM Credit in Fri Dec 01 14:49:00 WAT 2023 Amount 500
   ATM Debit  in Fri Dec 01 14:49:02 WAT 2023 Amount 100
   ATM Credit in Fri Dec 01 14:49:04 WAT 2023 Amount 500


                    Balance :-150DT

************************************************************
```

Aspect Programmatique

1. Implémentation du ATM

Différentes diverses fonctionnalités sont implémentées dans cette Applet

a) Hachage du Code PIN

Afin de stocker d'une manière sure le code PIN, on ne doit pas le garder sous sa forme original, un code PIN composé de 4 PIN.

Cependant on peut hacher le Code PIN et juste stocker le HASH.

Afin de vérifier si le code PIN entré est valide ou non, on doit juste hacher le code entré et comparé les deux hash. S'ils sont identiques, alors le code PIN est valide.

L'algorithme de hachage utilisé est le SHA-256 qui donne en sortie 32 octets (256 bits).

b) Utilisation des Transient_Array

Tout objet instancié avec le mot clé « new » est stocké par défaut dans le EEPROM qui est moi rapide de la RAM

Afin d'utiliser la RAM, on fait appel à « JCSystem.makeTransientArray() » afin d'instancier des tableaux dans la RAM. Ceci accélère le temps d'exécution de l'Applet.

c) Montant maximal autorisé

La bonne gestion de cette Applet a permis l'augmentation du solde maximal du client de 127 TND à plus que **2 000 000 TND** en instanciant la variable du solde comme int et gérer le calcul (incrémentation, décrémentation et initialisation) du solde par des tableaux.

d) Persistance des données

La fermeture de l'application ou la déconnexion de la carte ne résulte pas dans la perte des données.

Toute information sensible est stockée côté serveur afin d'assurer sa persistance. Votre solde n'est pas remis à zéro lors de la déconnexion de la carte ;

2. Extraits du code

a) Applet Java Card

Déclaration des constantes globales :

Code des instructions :

```
/* Code des instructions */
private static final byte CLA_MONAPPLET = (byte) 0xB0;
private static final byte INS_VERIF_PIN = 0x00;
private static final byte INS_INTERROGER_COMPTE = 0x01;
private static final byte INS_INCREMENTER_COMPTE = 0x02;
private static final byte INS_DECREMENTER_COMPTE = 0x03;
private static final byte INS_INITIALISER_COMPTE = 0x04;
```

Code des Exceptions

```
/* Exceptions */

// Verification Pin Echou
private final static short SW_VERIFICATION_FAILED = 0x6300;

// Nombre maximale d'essai atteint
private final static short SW_EXCEED_TRY_LIMIT = 0x6321;

// signal that the balance exceed the maximum
private final static short SW_EXCEED_MAXIMUM_BALANCE = 0x6A84;

// signal the the balance becomes negative
private final static short SW_NEGATIVE_BALANCE = 0x6A85;

// alerte que le compte est dans le rouge
private final static short SW_ROUGE=0x6320;
```

Code des constantes

```
/* Constants */
      // Solde Maximum permis
      private final static int MAX_BALANCE = 2001000;

      // Nombre Maximal de tentatives permises
      private final static byte MAX_ERROR_PIN = (byte) 0x03;

      // Longeur du code PIN
      private final static byte PIN_LENGTH = (byte) 0x04;

      // Montant maximale dans le rouge permis
      private final static int MAX_ROUGE=1000;

      // Longeur du tableau Transient
      private final static short TRANSIENT_BUFFER_SIZE = (byte) 127;
```

Les Variables

```
/* variables */

	// Solde de compte
	private static int balance ;

	//Nombre d'essai du code pin restants
	private byte remainingTries;

	// Tableau Transient pour stocker les valeurs intermédiaires
	private byte[] transition_buffer;

	// Stocker la hash du Code PIN
	private byte[] hashedPIN;

	// Objet utilisé pour le hash
	private MessageDigest sha256;
```

Implémentation du constructeur

Cette méthode instancie les variables précédemment déclarées et les initialises

```
private Bank(byte[] bArray,int i,int j) {
	balance = MAX_ROUGE;
	transition_buffer = JCSystem.makeTransientByteArray(
						(short)TRANSIENT_BUFFER_SIZE,
						JCSystem.CLEAR_ON_RESET
	);
		remainingTries = MAX_ERROR_PIN;
		//intilaize hash object
		sha256 = MessageDigest.getInstance(
					MessageDigest.ALG_SHA_256, false
		);
		// Allocate memory for HashedPIN and initialize it with the PIN's
hash
		hashedPIN = new byte[] {
(byte) 0x27, (byte) 0xEC, (byte) 0xD0, (byte) 0xA5, (byte) 0x98, (byte) 0xE7,
(byte) 0x6F, (byte) 0x8A, (byte) 0x2F, (byte) 0xD2, (byte) 0x64, (byte) 0xD4,
(byte) 0x27, (byte) 0xDF, (byte) 0x0A, (byte) 0x11, (byte) 0x99, (byte) 0x03,
(byte) 0xE8, (byte) 0xEA, (byte) 0xE3, (byte) 0x84, (byte) 0xE4, (byte) 0x78,
(byte) 0x90, (byte) 0x25, (byte) 0x41, (byte) 0x75, (byte) 0x6F, (byte) 0x08,
(byte) 0x9D, (byte) 0xD1
		}; //hash for code 1111

}
```

Redéfinitions de la méthode Select()

Ne pas sélectionner l'Applet si la carte est désactivée

```
public boolean select() {
             // pas de selection si le pin est blocker
             if (remainingTries == 0)
                    return false;
             return true;
      }
```

Redéfinitions de la méthode deselect()

Réinitialiser le nombre d'essais

```
public void deselect() {
          remainingTries = MAX_ERROR_PIN;
      }
```

Redéfinitions de la méthode process()

```
public void process(APDU apdu) {
             // Lire le buffer
             byte[] buffer = apdu.getBuffer();
             // exception qui teste sur la commande de selection
             if (apdu.isISOInterindustryCLA()) {
                    if (buffer[ISO7816.OFFSET_INS] == (byte) (0xA4)) {
                           return;
                    } else {
                           ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
                    }
             }
                    // Vérifier si l'APDU reçu est celui de selection
             if (this.selectingApplet())
                    return;
             if (buffer[ISO7816.OFFSET_CLA] != CLA_MONAPPLET) {
                    ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
             }
             switch (buffer[ISO7816.OFFSET_INS]) {
             case INS_VERIF_PIN:
                    verifyPIN(apdu);
                           break;
             case INS_INCREMENTER_COMPTE:
                    credit(apdu);
                    break;
             case INS_DECREMENTER_COMPTE:
                    debit(apdu);
                    break;
```

```
case INS_INTERROGER_COMPTE:
                    getBalance(apdu);
                    break;
             case INS_INITIALISER_COMPTE:
                    setBalance(apdu);
                    break;
             default:
                    ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
             }
      }
```

Implémentation de méthode credit()

```
private void credit(APDU apdu) {


             byte[] buffer = apdu.getBuffer();

             // Lc byte denotes the number of bytes in the
             // data field of the command APDU
             byte numBytes = buffer[ISO7816.OFFSET_LC];

             // indicate that this APDU has incoming data
             // and receive data starting from the offset
             // ISO7816.OFFSET_CDATA following the 5 header
             // bytes.
             byte byteRead = (byte) (apdu.setIncomingAndReceive());

             // it is an error if the number of data bytes
             // read does not match the number in Lc byte
             if ((numBytes != 2) || (byteRead != 2))
                    ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

             // get the credit amount
             short creditAmount = Util.getShort(buffer,
ISO7816.OFFSET_CDATA);

             // check the new balance
             if ( (balance + creditAmount) > MAX_BALANCE)
                    ISOException.throwIt(SW_EXCEED_MAXIMUM_BALANCE);

             // credit the amount
             balance = (balance + (int) creditAmount);
      }
```

Implémentation la méthode debit()

```
private void debit(APDU apdu) {

		byte[] buffer = apdu.getBuffer();

		byte numBytes = (byte) (buffer[ISO7816.OFFSET_LC]);

		byte byteRead = (byte) (apdu.setIncomingAndReceive());

		if ((numBytes != 2) || (byteRead != 2))
			ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

		// Get debit amount
		short debitAmount = Util.getShort(buffer,
ISO7816.OFFSET_CDATA);

		// check the new balance
		if ((balance - debitAmount) < 0)
			ISOException.throwIt(SW_NEGATIVE_BALANCE);

		balance = (balance - (int) debitAmount);

		if (balance <MAX_ROUGE)
			ISOException.throwIt(SW_ROUGE);
	}
```

Implémentation de la méthode getBalance()

```
	private void getBalance(APDU apdu) {
		byte[] buffer = apdu.getBuffer();

		short le = apdu.setOutgoing();

		if (le < 2)
			ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

		apdu.setOutgoingLength((byte) 3);

		//Byte shifting pour diviser la valeur sur les différentes
		// cases du tableau
		buffer[0] = (byte) ((balance >> 16) & 0xFF);
		buffer[1] = (byte) ((balance >> 8) & 0xFF);
		buffer[2] = (byte) (balance & 0xFF);

		apdu.sendBytes((short) 0, (short) 3);
	}
```

Implémentation de la méthode setBalance()

```
private void setBalance(APDU apdu) {
		byte[] buffer = apdu.getBuffer();

		byte numBytes = buffer[ISO7816.OFFSET_LC];

		byte byteRead = (byte) (apdu.setIncomingAndReceive());

		if (
			(numBytes != TRANSIENT_BUFFER_SIZE) ||
			(byteRead != TRANSIENT_BUFFER_SIZE)
		)
			ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

		Util.arrayCopyNonAtomic(
					buffer,
					ISO7816.OFFSET_CDATA,
					transition_buffer,
					(short) 0,
					byteRead
		);

		// Formule de calcul: (a + b + ... + z) * 127 + reste avec a, _ , z
les chiffres passés dans le buffer
		// de la case ISO7816.OFFSET_CDATA jusqu'à ISO7816.OFFSET_CDATA +
(byteRead - 1)
		int a_z=0;
		for(short i= (0); i < byteRead - 1 ; i++) {
			a_z += transition_buffer[i];
		}
		balance = a_z * 127 + transition_buffer[byteRead-1];
	}
```

Implémentation de la méthode verifyPIN()

```
private void verifyPIN(APDU apdu) {
		byte[] buffer = apdu.getBuffer();
		//byte byteRead = (byte) (apdu.setIncomingAndReceive());
		if(remainingTries==(byte) 1)
			ISOException.throwIt(SW_EXCEED_TRY_LIMIT);
		remainingTries--;
		//Check the resemblance of the hashed PIN and the stored PIN
		if(verifyPINHash(
					buffer,
					(short) ISO7816.OFFSET_CDATA,
					PIN_LENGTH) == false
		)
			ISOException.throwIt(SW_VERIFICATION_FAILED);
	}
```

Implémentation de la méthode setPIN()

```
private void setPIN(byte[] buffer, short offset, short length) {
        // Perform PIN hashing and store the hashed
        hashAndStorePIN(buffer, offset, length);
    }
```

Implémentation de la méthode hashAndStorePIN()

```
private void hashAndStorePIN(byte[] pinBuffer, short pinOffset,
short pinLength) {
       // Hash the PIN using SHA-256
       sha256.reset();
       sha256.doFinal(pinBuffer, pinOffset, pinLength, hashedPIN,
(short) 0);
    }
```

Implémentation de la méthode verifyPINHash()

```
private boolean verifyPINHash(byte[] pinBuffer, short pinOffset, short
pinLength) {
        byte[] inputHash = JCSystem.makeTransientByteArray((short) 32,
JCSystem.CLEAR_ON_RESET);
        // Hash the provided PIN
        sha256.reset();
        sha256.doFinal(pinBuffer, pinOffset, pinLength, inputHash,
(short) 0);
        // Compare the hashed PINs and return the result
        return Util.arrayCompare(inputHash, (short) 0, hashedPIN,
(short) 0, (short) 32) == 0;
    }
```

b) BankFunction.java

```
public class BankFunction {

       private final static byte INS_GET_BALANCE = (byte) 0x01;
       private final static byte INS_INITIALISER_COMPTE = (byte)
0x04;
    static Apdu apdu ;
    static CadT1Client cad;
    public static int balance;
    private static String FILE_PATH = "Balance.txt";
    private static boolean notCreated=true;
```

```
    public BankFunction() {
       try {
       this.balance = readFromFile(FILE_PATH);
       }
    catch(IOException | NumberFormatException e) {
             createFile ("Balance.txt","1000");
             this.balance=1000;
    }


    }
    public Apdu Msg(byte ins, byte lc, byte[] data,byte le) throws IOException,
CadTransportException{
        apdu = new Apdu();
        apdu.command[Apdu.CLA] = (byte) 0xB0;
        apdu.command[Apdu.P1] = 0x00;
        apdu.command[Apdu.P2] = 0x00;
        apdu.command[Apdu.INS] = ins;
        //apdu.setLe(0x7f);
        apdu.setLe(le);
        if (data!=null)
            apdu.setDataIn(data);
        cad.exchangeApdu(apdu);
        return apdu;
    }


    void updateBalance() throws IOException, CadTransportException {
       Apdu apdu = null;
       // transition_buffer Size defined in Bank.java (JavaCard Applet)
       byte size = 127;
       // Data Array transfered in APDU message
       byte[] data = new byte[size];
       // Formule: (a + b + ... + z)*127+rest
       int a_z;
       byte rest;
       rest = (byte) ( balance % 127);
       a_z = (balance) / 127;
       for(byte i= 0; i<size - 1 ; i++) {
             if(a_z > 127) {
                   data[i] = 127;
                   a_z -= 127;
             }
             else {
                   data[i] = (byte) a_z;
                   a_z = 0;
             }

       }
       data[size-1] = rest;

       apdu = Msg(INS_INITIALISER_COMPTE, (byte) size, data, (byte) 0x7F);
    }
```

```
   public void Connect(){
        Socket sckCarte;

        try {
            sckCarte = new Socket("localhost", 9025);
            sckCarte.setTcpNoDelay(true);
            BufferedInputStream input = new
BufferedInputStream(sckCarte.getInputStream());
            BufferedOutputStream output = new
BufferedOutputStream(sckCarte.getOutputStream());
            cad = new CadT1Client(input, output);
        } catch (Exception e) {
            System.out.println("Erreur : impossible de se connecter a la
Javacard");
            return;
        }
        /* Mise sous tension de la carte */
        try {
            cad.powerUp();
        } catch (Exception e) {
            System.out.println("Erreur lors de l'envoi de la commande Powerup a
la Javacard");
            return;
        }
    }

    public void select() throws IOException, CadTransportException{

        /* Sélection de l'applet :création du commande SELECT APDU */
        apdu = new Apdu();
        apdu.command[Apdu.CLA] = (byte) 0x00;
        apdu.command[Apdu.INS] = (byte) 0xA4;
        apdu.command[Apdu.P1] = 0x04;
        apdu.command[Apdu.P2] = 0x00;
        byte[] appletAID = { 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
0x09, 0x00, 0x00 };
        apdu.setDataIn(appletAID);
        cad.exchangeApdu(apdu);
        if (apdu.getStatus() != 0x9000) {
            System.out.println("Erreur lors de la sélection de l'applet");
            System.exit(1);
        }

    }
```

```
public void Deselect(){
        /* Mise hors tension de la carte */
        try {
             getBalance();
             int balance_what = this.balance;
             writeToFile(FILE_PATH, balance);
            cad.powerDown();
        } catch (Exception e) {
            System.out.println("Erreur lors de l'envoi de la commande Powerdown
a la Javacard");
            return;
        }
    }
    // Method to write a number to the file
    private static void writeToFile(String filePath, int number) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath))) {
            // Write the number to the file
            writer.write(Integer.toString(number));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    private static void writeToFile(String filePath, String data,boolean b) {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath,b))) {
            // Write the number to the file
            writer.write(data);
            writer.newLine();
        } catch (IOException e) {
             e.printStackTrace();
        }
    }



    public static void createTransactionFile (String filePath) {
       createFile(filePath,"");

       writeToFile(filePath,"************************************************
*********",false);

       writeToFile(filePath,"                    Transaction Ticket",true);
       writeToFile(filePath,"",true);


    }

    public static void addToTransactionFile (String data) {
       if (notCreated) {
             createTransactionFile("Transaction.txt");
             notCreated=false;
       }
       writeToFile("Transaction.txt",data,true);

    }
```

```
// Method to read a number from the file
    private static int readFromFile(String filePath)throws
IOException,NumberFormatException {
        int number = 0;
        BufferedReader reader = new BufferedReader(new FileReader(filePath));
            // Read the first line from the file
            String line = reader.readLine();

            // Convert the content to an integer
            number = Integer.parseInt(line);

        return number;
    }

    private static void createFile (String filePath,String data) {

        // Create a File object
        File file = new File(filePath);

        try {
            // Create the file
            file.createNewFile();
            FileWriter writer = new FileWriter(file);
            BufferedWriter bufferedWriter = new BufferedWriter(writer);
            bufferedWriter.write(data);
            bufferedWriter.close();
            writer.close();
        } catch (IOException e) {
            System.err.println("Error creating the file: " + e.getMessage());
        }
    }



    private void getBalance() throws IOException, CadTransportException {
       Apdu apdu=null;
             apdu = Msg(INS_GET_BALANCE, (byte) 0x00, null, (byte) 0x7f);
             BigInteger one= new BigInteger(apdu.dataOut);
        balance = one.intValue();
             System.out.println();
             System.out.println(apdu);

    }


}
```