

République Tunisienne

Ministre de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Monastir

Institut Supérieur d'Informatique et de Mathématiques de Monastir

Département Informatique



Rapport de Projet

Présenté dans le cadre de la matière

Application Logicielle d'Entreprise

Spécialité

Ingénierie en Informatique Génie Logiciel

Réalisé par

Yassine El Kamel
Dhia Ben Hamouda

Encadré par

Dr. Ramzi Mahmoudi

Mise en place d'une plateforme de
Distributeur d'argent automatique

Table des matières

Installation du JDK.....	6
Setup du JCDK:	6
Ajouter les variables d'environnement nécessaires.....	7
A. JAVA_HOME:.....	7
B. JC_HOME.....	9
Vérifier la bonne installation du JCDK.....	9
Vérifier la bonne installation du JavaCard	9
Ajouter les librairies(API) JavaCard à votre environnement de développement	10
Création du premier projet JavaCard.....	14
Conclusion :.....	15
Création de notre Projet JavaCard.....	17
Création de notre première Applet JavaCard	19
Implémentation de notre Applet.....	21
A. Importer l'API JavaCard.....	21
B. Déclarer les attributs et les constantes.....	21
C. Définition des méthodes publiques qu'on doit implémenter	22
D. Outils de simulation	23
Sans conservation d'état.....	25
Concept	25
TEST	29
Avec Conservation d'état.....	30
Concept :	30
TEST :	30
Création de l'application client sous Eclipse :	36
A. Création d'un nouveau projet :	36
B. Ajout de la librairie « apduio » dans le classpath :	37
C. Création de la classe principale :	40
Les importations:	42
Étape 1 - Connexion :	43
Etape 2 Sélection :.....	43
Etape 3 - Invocation des services implémentés :.....	44

D. Utilisation de l'application cliente avec un simulateur – JCWDE :	47
Interface Graphique	53
A. Vérification du Code PIN	53
B. Menu Solde	56
C. Le Menu Solde affiche le solde de votre compte	56
D. Menu Créditer	57
E. Menu Débitier	59
F. Menu Imprimer mes Transactions	61
Aspect Programmatique	63
A. Implémentation du ATM	63
a) Hachage du Code PIN	63
b) Utilisation des Transient_Array	63
c) Montant maximal autorisé	63
d) Persistance des données	63
B. Extraits du code	64
a) Applet Java Card	64
Déclaration des constantes globales :	64
Code des instructions :	64
Code des Exceptions	64
Code des constantes	64
Les Variables	65
Implémentation du constructeur	65
Redéfinitions de la méthode Select()	66
Redéfinitions de la méthode deselect()	66
Redéfinitions de la méthode process()	66
Implémentation de méthode credit()	67
Implémentation la méthode debit()	68
Implémentation de la méthode getBalance()	68
<i>Implémentation de la méthode setBalance()</i>	69
Implémentation de la méthode verifyPIN()	69
Implémentation de la méthode setPIN()	70
Implémentation de la méthode hashAndStorePIN()	70

Implémentation de la méthode verifyPINHash()	70
b) BankFunction.java	70

Compte Rendu Manip-01

Préparation de l'Environnement de développement JavaCard

Yassine El Kamel

Dhia Ben Hamouda

ING INFO 1 TD1

Installation du JDK

Accéder au site <https://www.oracle.com/java/technologies/downloads/>

JDK 21	<u>JDK 17</u>	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	<u>Windows</u>	
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.

Setup du JCDK:

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

B) On choisit 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

Ajouter les variables d'environnement nécessaires

On ajoutera dans cette parties deux variables d'environnements

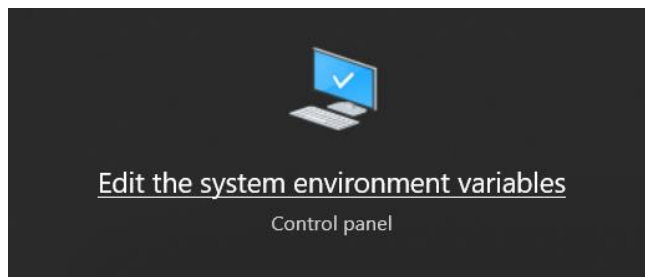
- JAVA_HOME

-JC_HOME

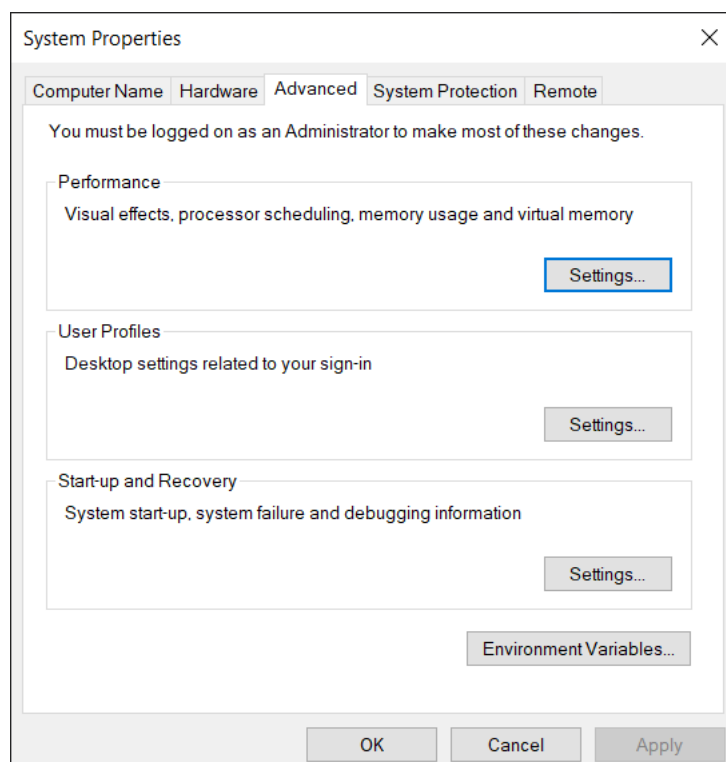
Ceci est pour rendre au system découvrable les exécutables du JDK et du JCRE (JavaCard RunTime Environment)

A.JAVA_HOME :

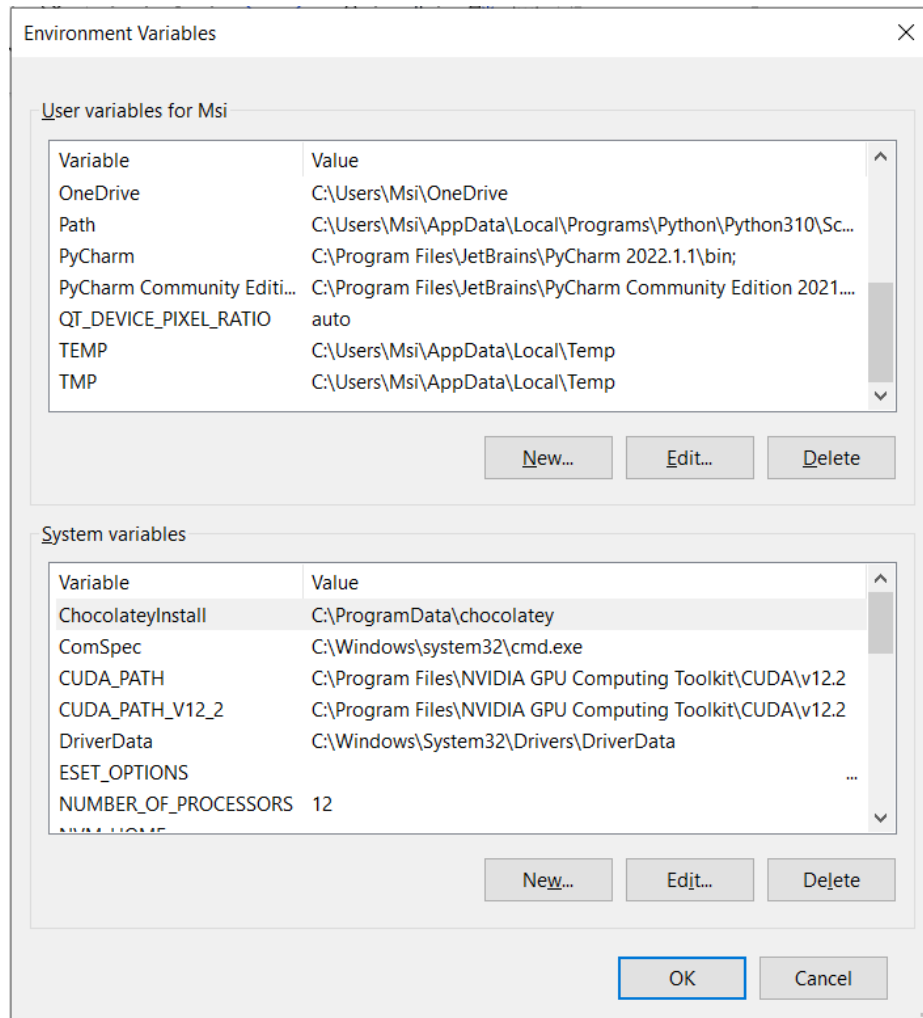
-Accéder au variable d'environnement :



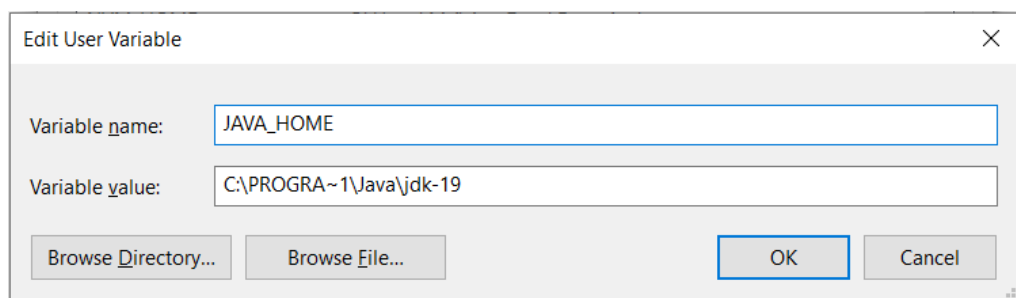
-Appuyer sur variables d'environnement :



-Appuyer sur nouveau



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



Remarque : 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 racine suivante « %JAVA_HOME%\bin »

`%JAVA_HOME%\bin`

B. JC_HOME

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

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

-JC_HOME_v2

-JC_HOME_v3

-JC_HOME

Afin 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

Vérifier la bonne installation du JCDK

Entrer la commande «java -version » est un message similaire à 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)
```

Vérifier la bonne installation du JavaCard

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

(JavaCard 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
```

(JavaCard v3)

```
C:\Users\Msi>apdutool
ApuTool [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

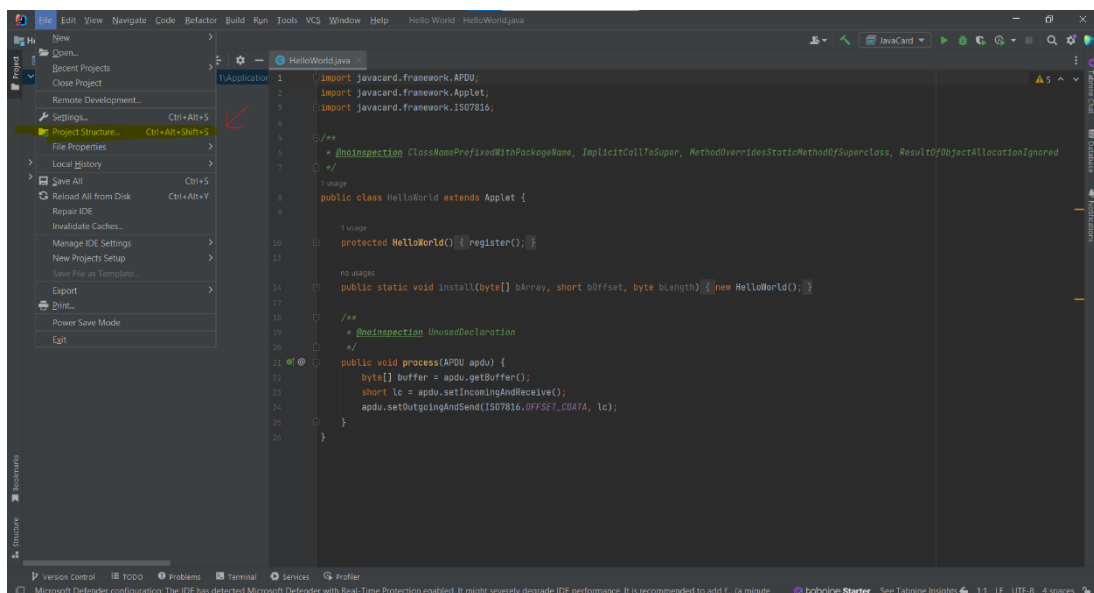
Ajouter les librairies (API) JavaCard à votre environnement de développement

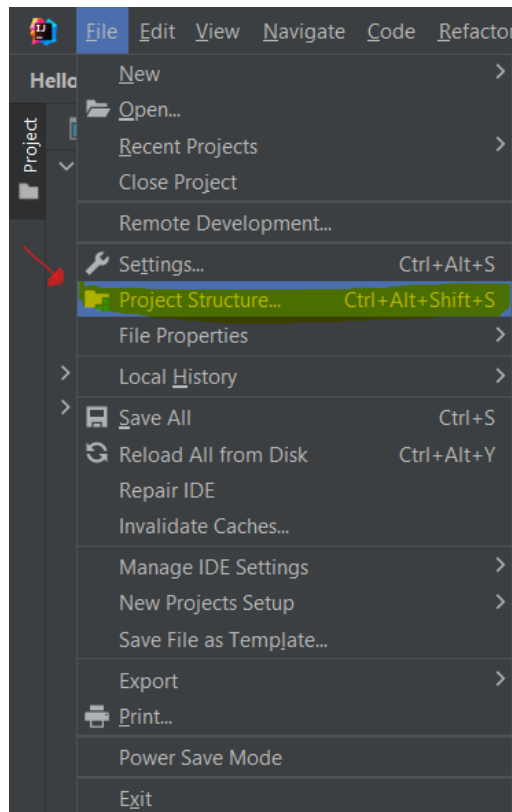
Pour les utilisateurs IntelliJ :

A) Télécharger un Template JavaCard de GitHub (lien :

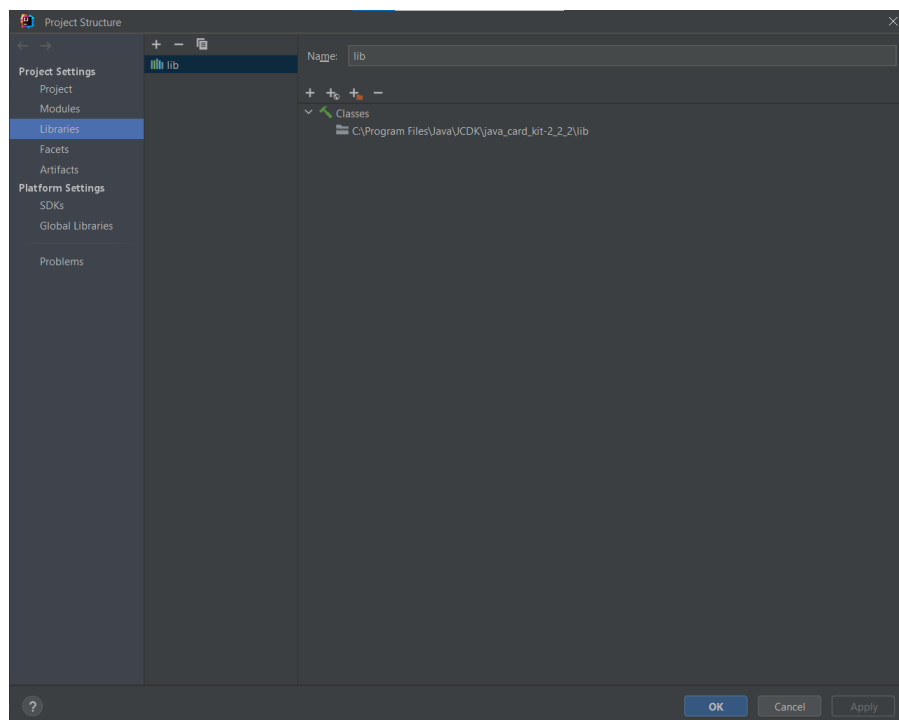
<https://github.com/FractalizeR/IntelliJ-IDEA-Java-Card-Project-Template.git>)

B) Accédez au « File -> Project Structure »





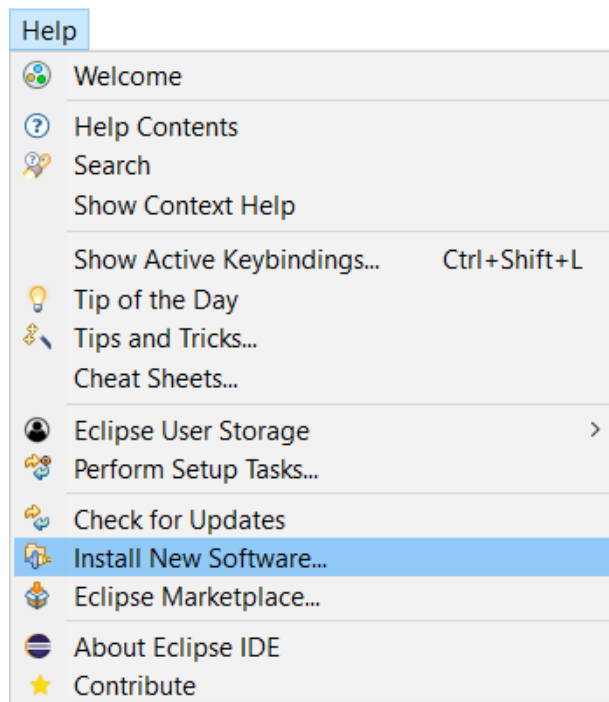
- C) Tapez sur le menu Librairies
- D) Cliquer sur l'icône +
- E) Parcourir le dossier C:\JavaCard
- F) Ajouter le sous-dossier «lib »



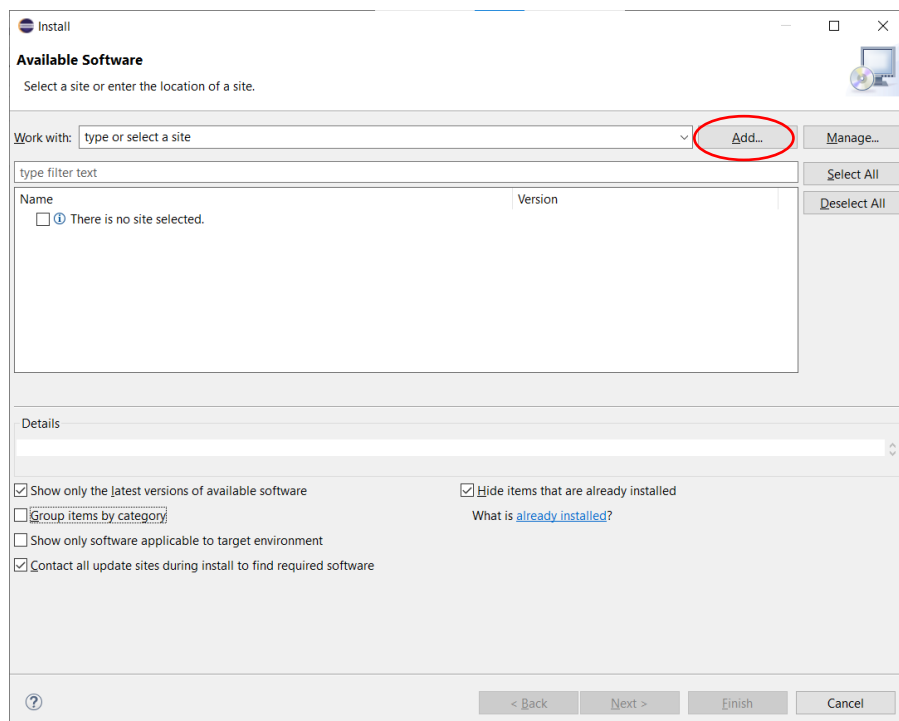
- G) Cliquez sur confirmer

Pour les utilisateurs Eclipse (JavaCard v3.x) :

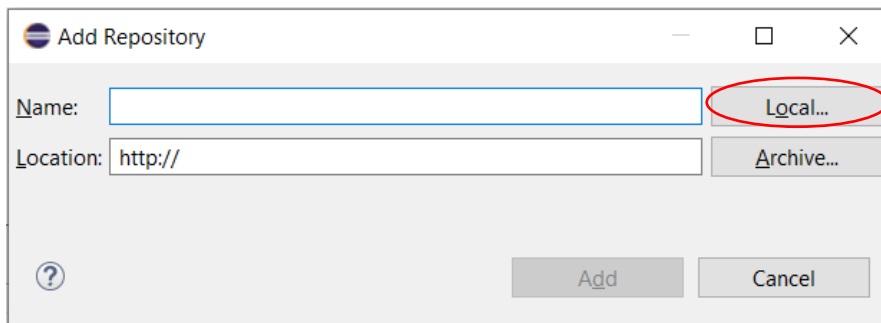
A) Accéder au menu « HELP » et à « Install New Software »



B) Taper sur « Ajouter »



C) Cliquer sur « Archive »

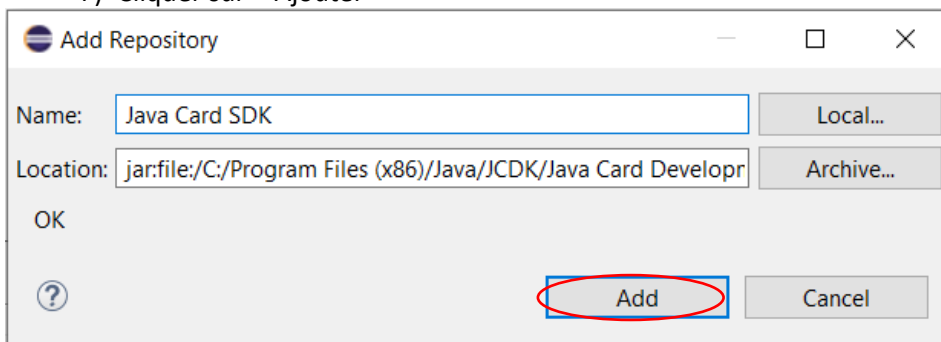


D) Sélectionner le kit de développement Eclipse plug-in dans :

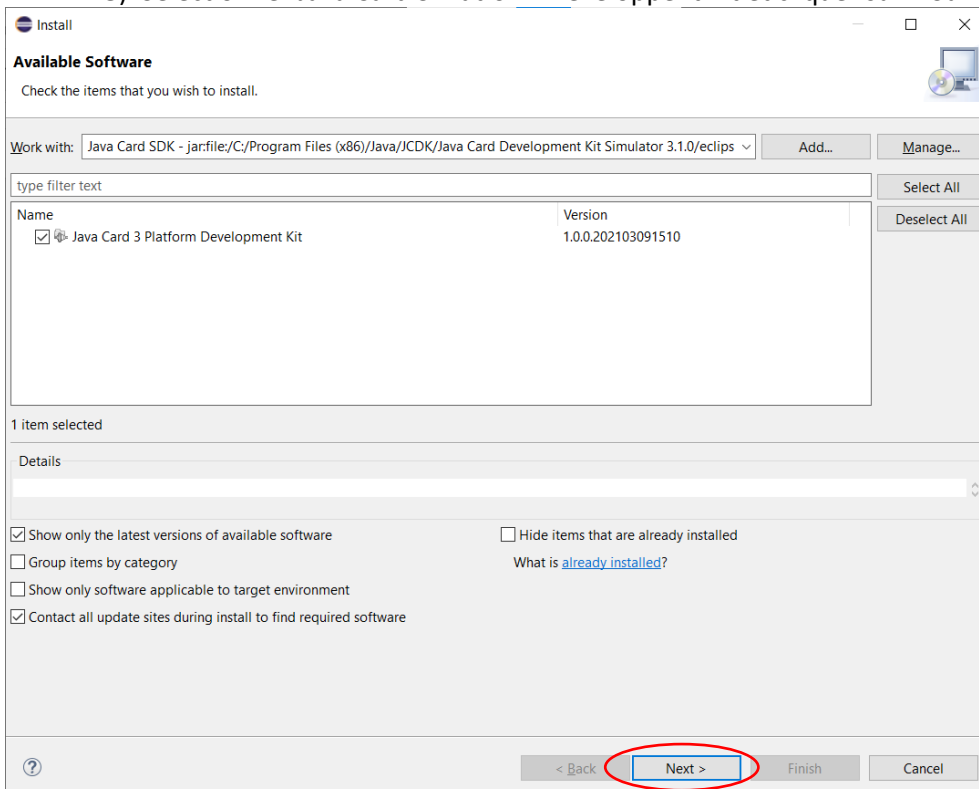
JC_HOME\eclipse-plugin\jcdk-repository_yyyymmddxxxx.zip

E) Ajouter comme Nom : Java Card SDK

F) Cliquer sur « Ajouter »



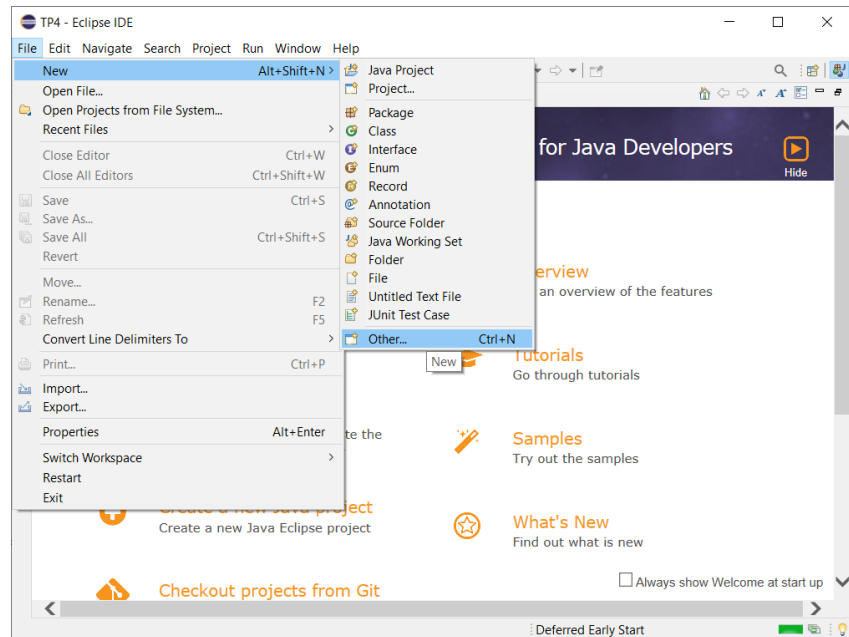
G) Sélectionner Java Card 3 Platform Développement Kit et cliquer sur « Suivant »



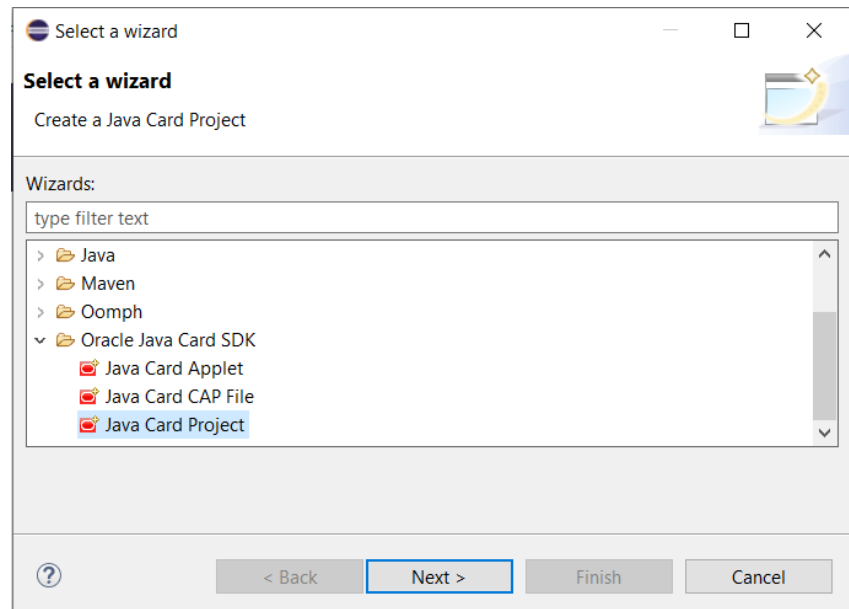
H) Attendre l'achèvement de l'installation et redémarrer Eclipse quand il vous le demande

Création du premier projet JavaCard

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



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



C) Cliquer sur 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.



Compte Rendu Manip-02

Développement d'une Applicatoïn côté serveur

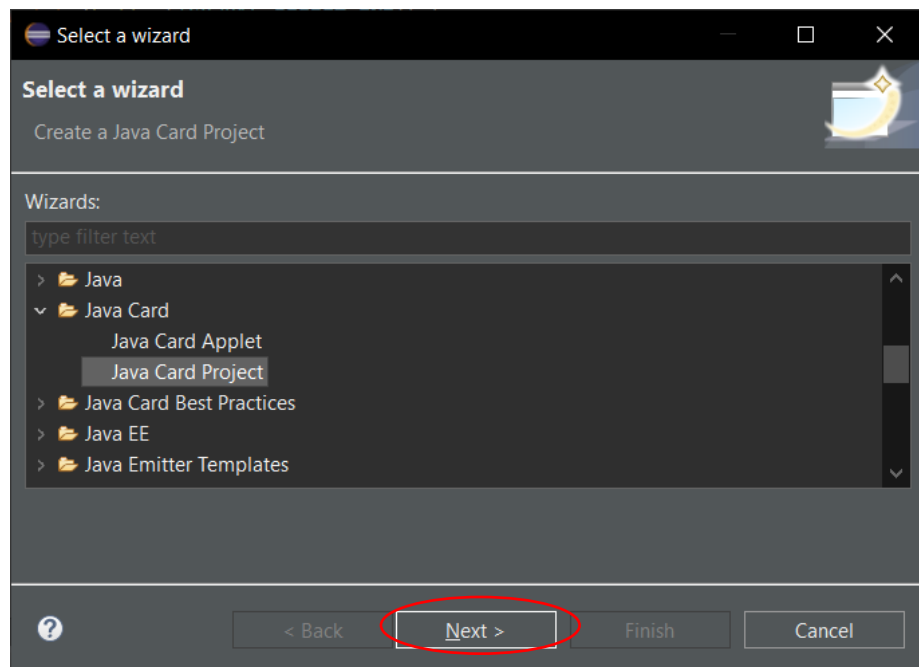
Yassine El Kamel

Dhia Ben Hamouda

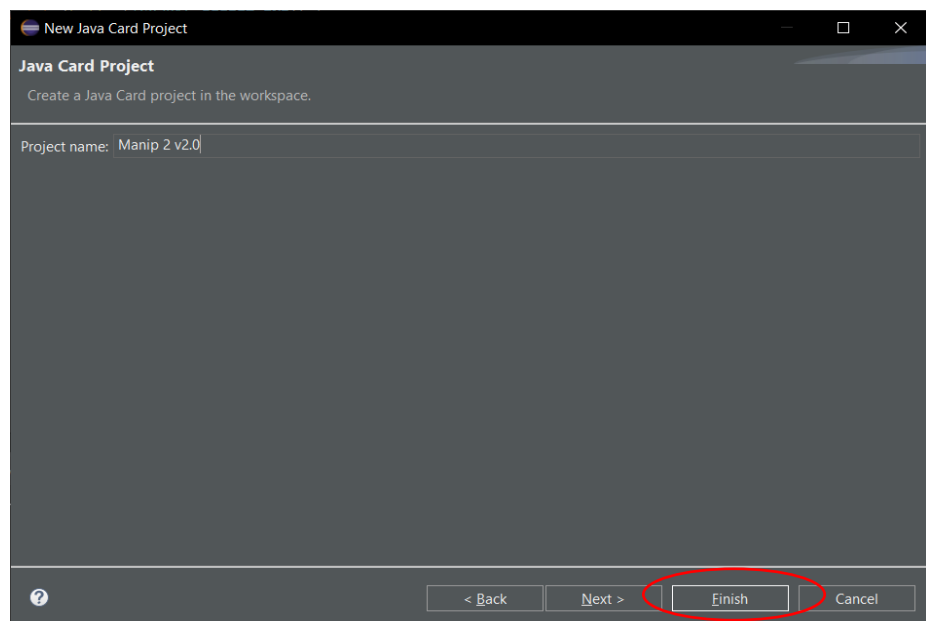
ING INFO 1 TD1

Création de notre Projet JavaCard

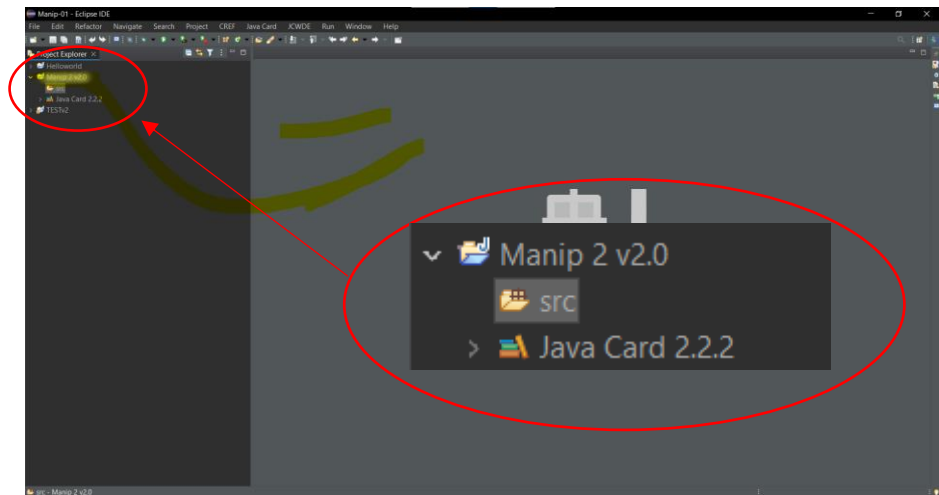
On Créer notre premier Projet JavaCard



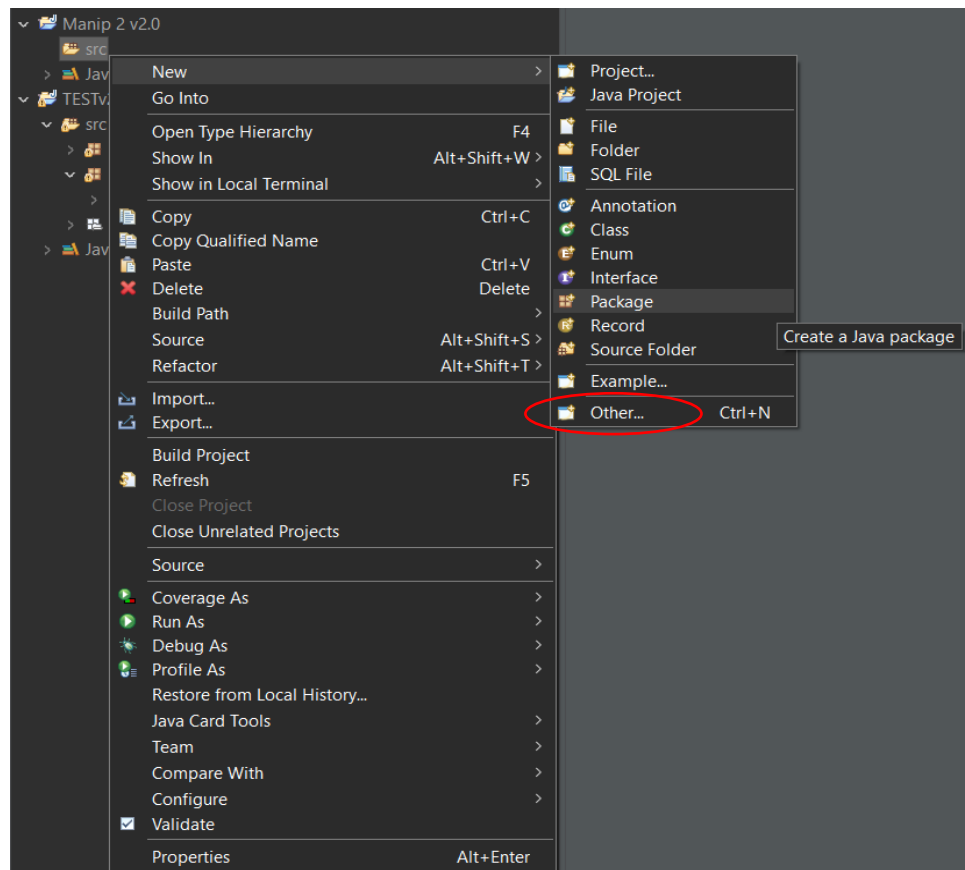
On nomme notre Premier Projet



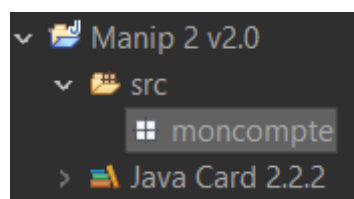
Maintenant on dispose de notre premier projet



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

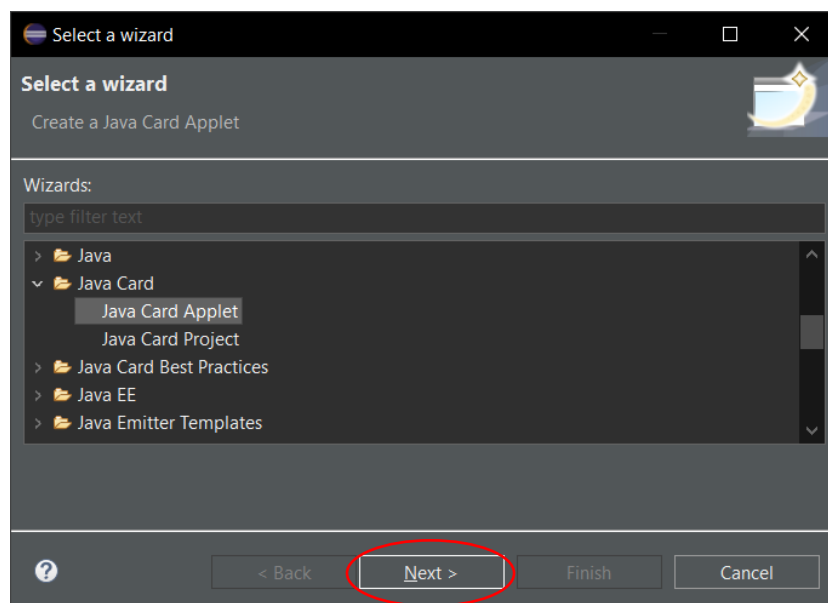
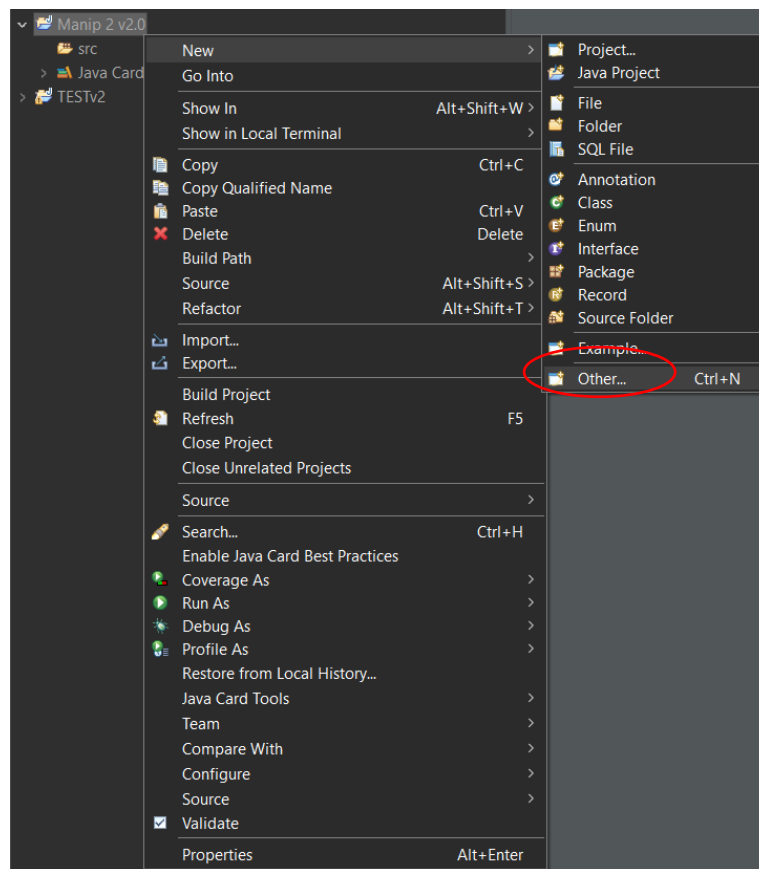


Etat Final :



Création de notre première Applet JavaCard

Clic droit sur notre projet -> Nouveau -> Autres...



New Java Card Applet

Create a new Java Card Applet class.

Source folder: Manip 2 v2.0/src Browse...

Package: moncompte Browse...

☐ Enclosing type: Browse...

Applet AID: 0x01:0x02:0x03:0x04:0x05:0x06:0x07:0x08:0x09:0x00:0x00

Name: MonApplet

Modifiers: ☐ public ☒ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: javacard.framework.Applet Browse...

Interfaces: Add...
Remove

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

? < Back Next > **Finish** Cancel

Eclipse nous génère alors le squelette de notre Applet automatiquement

```

1 package moncompte;
2
3 import javacard.framework.*;
4
5 public class MonApplet extends Applet {
6
7     private MonApplet() {
8     }
9
10    public static void install(byte[] bArray, short bOffset, byte bLength) throws ISOException {
11        new MonApplet().register();
12    }
13
14    public void process(APDU arg) throws ISOException {
15        // TODO Auto-generated method stub
16    }
17
18    }
19
20 }
21
22

```

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'identifier la famille 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 reconnaître quelle méthode on souhaite exécuter

```
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.

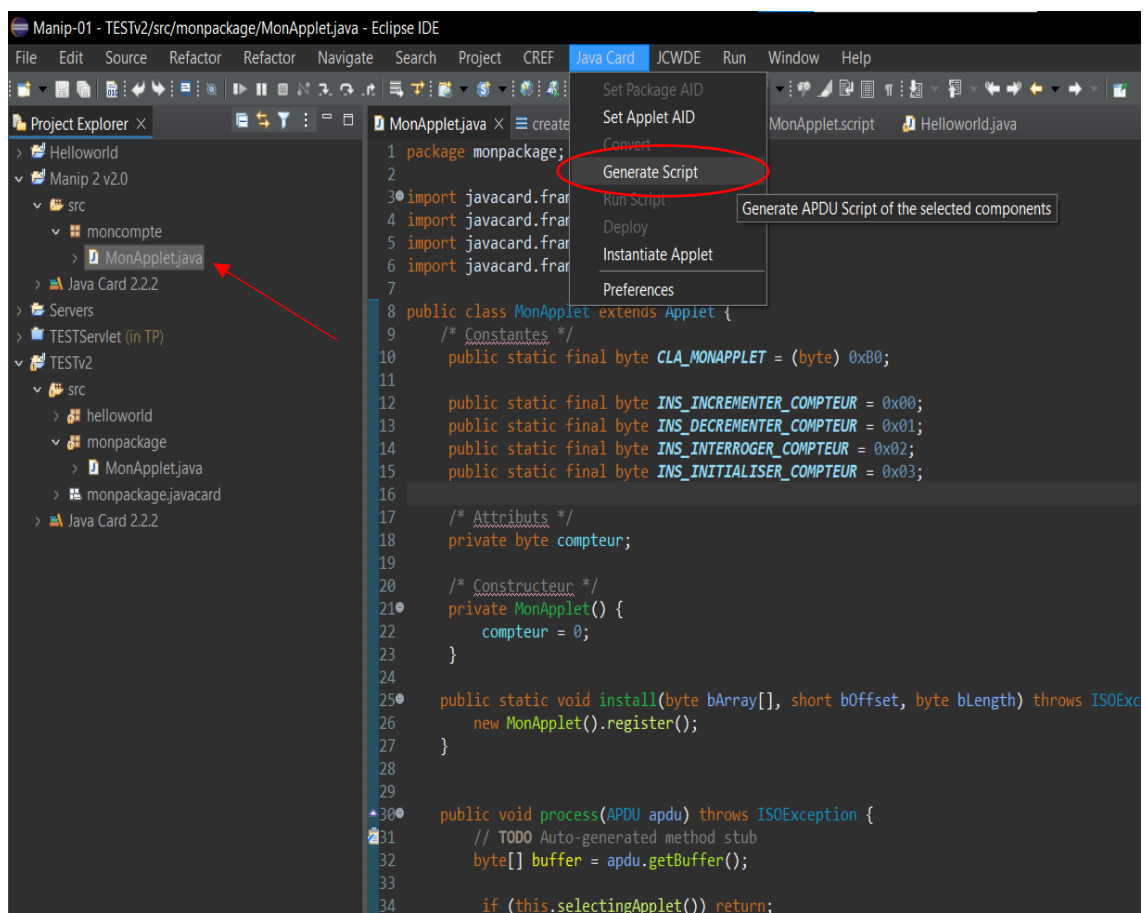
Ensuite il se renseigne sur l'identifiant de la méthode sélectionner et l'exécute.(Le switch case) tout en gérant les exceptions.

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 :

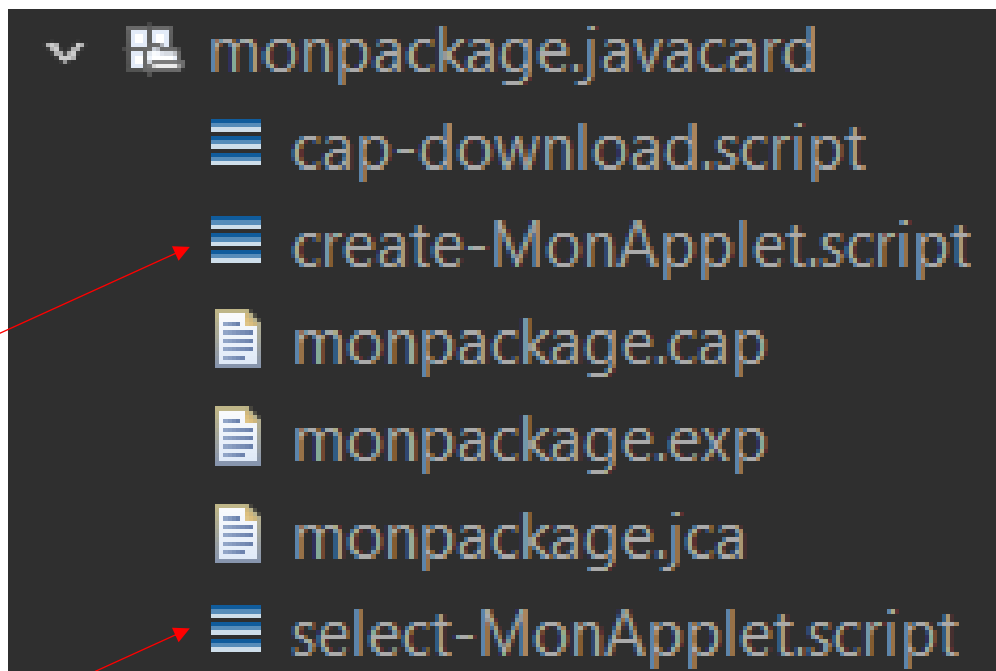
Notre Applet -> Menu JavaCard -> Générer Script



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:



On doit prendre en compte ces outils de simulation :

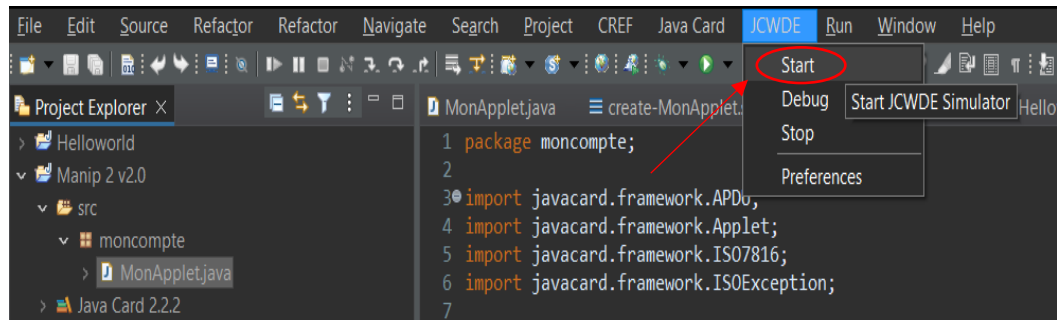
- APDUTOOL : Il gère l'échange des APDUs entre l'applet et la carte JavaCard (virtuelle ou réelle)
- JCWDE (Java Card Workstation Development Environment)
:_Programme qui simule le JCRE dans une JVM
- CREF (Java Card Platform Simulator):

Sans conservation d'état

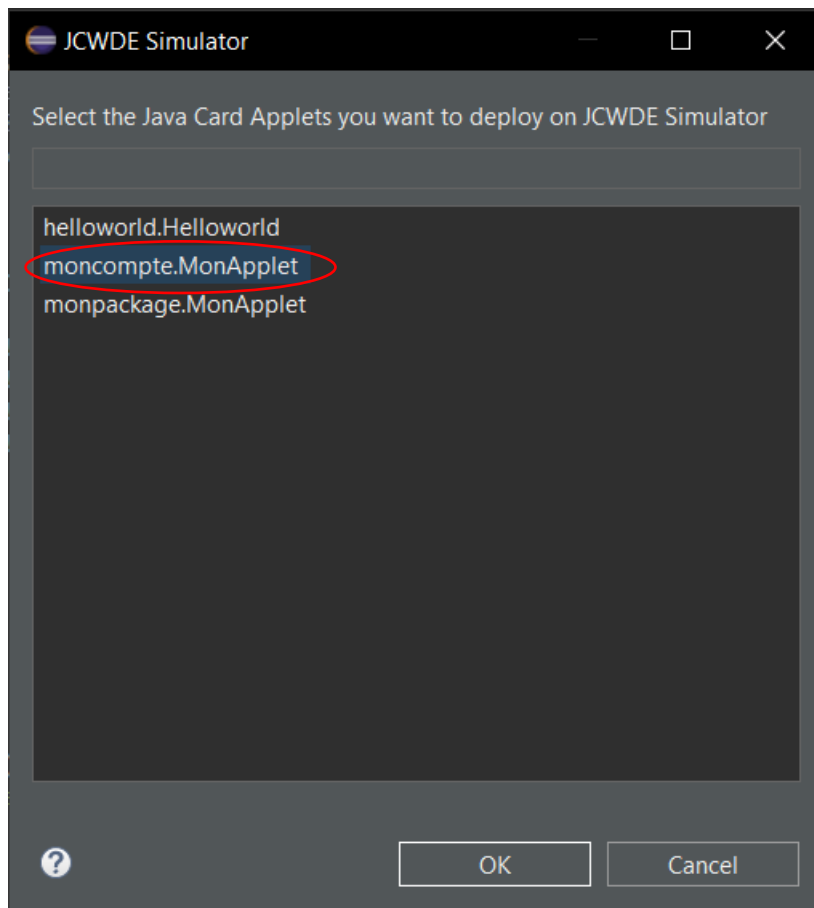
Concept

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
```



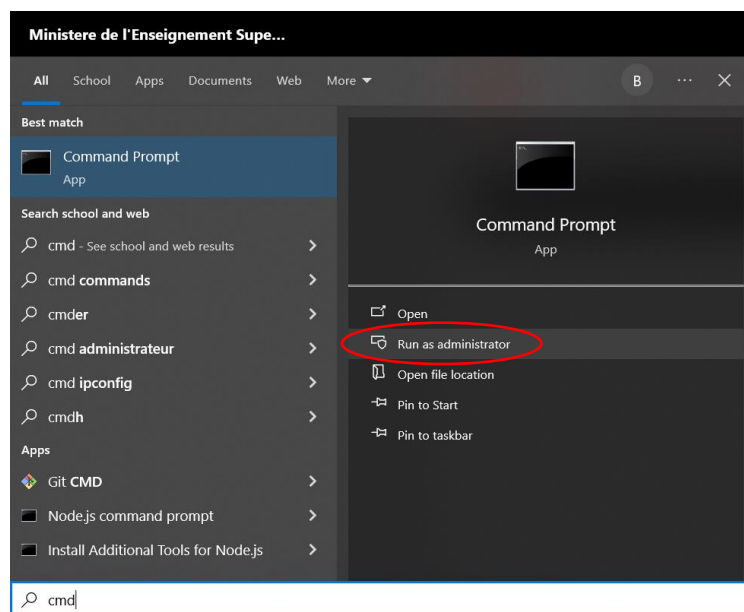
- Si vous rencontrez un message d'erreur comme ci-dessous et que vous êtes sûr d'avoir bien installé JavaCard sur votre machine, alors essayer de lancer l'invite de commande (CMD) en tant qu'Administrateur

Erreur :

```
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Msi>apdutool
'apdutool' is not recognized as an internal or external command,
operable program or batch file.
```

Solution :



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>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
```

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 X
1powerup;
2// Select the installer applet
30x00 0xA4 0x04 0x00 0x00 0xa0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
4// create MonApplet applet
50x80 0xB8 0x00 0x00 0xd 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x00 0x7F;
```

```
select-MonApplet.script X
1powerup;
2// select MonApplet applet
30x00 0xA4 0x04 0x00 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x7F;
```

Aussi taper la commande « powerup » une seul fois



- 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 Hexadécimal
- On concatène les deux champs pour avoir le code message
- Le code de réussite 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 0x00 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 installé et sélectionner notre Applet et appuyer sur entrer après.

TEST

Pour tester notre applet on va lancer les commandes suivantes :

- 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 décrémenté

```
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

Avec Conservation d'état

Pour ceci on va utiliser l'outil CREF.

Concept :

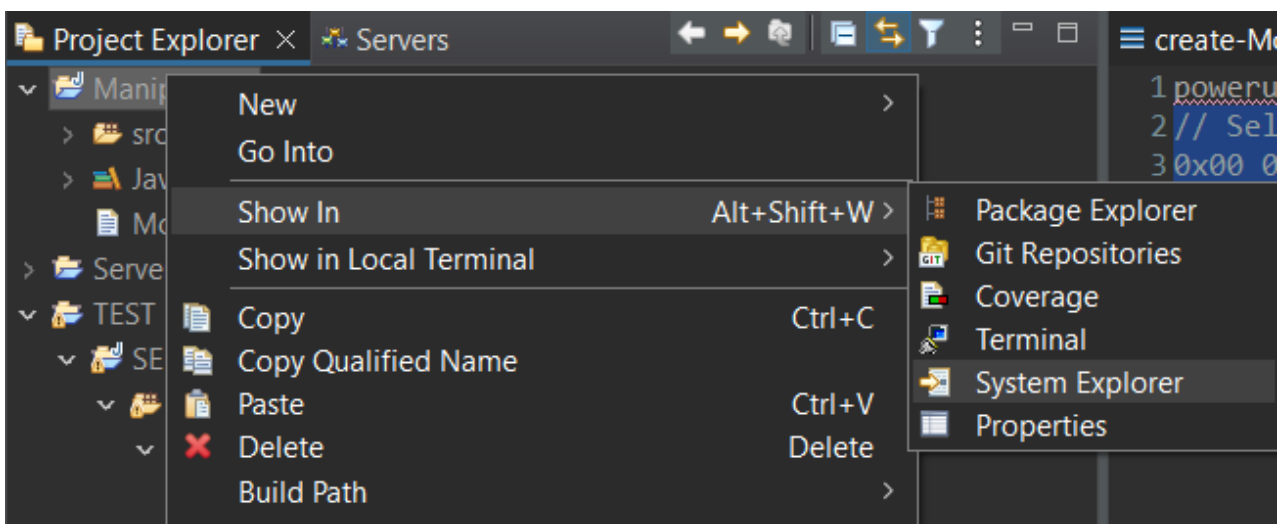
Pour Ceci on suivra les étapes suivantes :

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

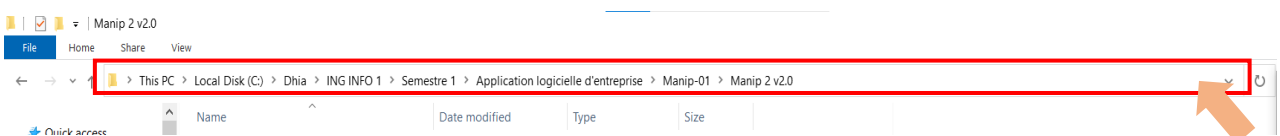
TEST :

Tout d'abord on crée notre fichier eeprom :

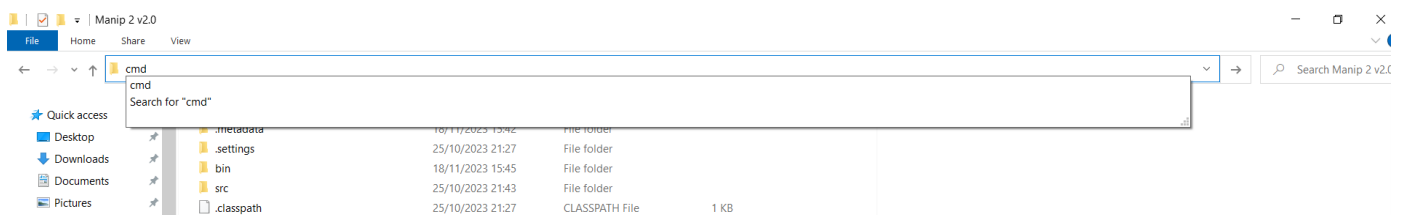
- On ouvre le dossier de notre projet JavaCard



- On clique sur la barre de chemin

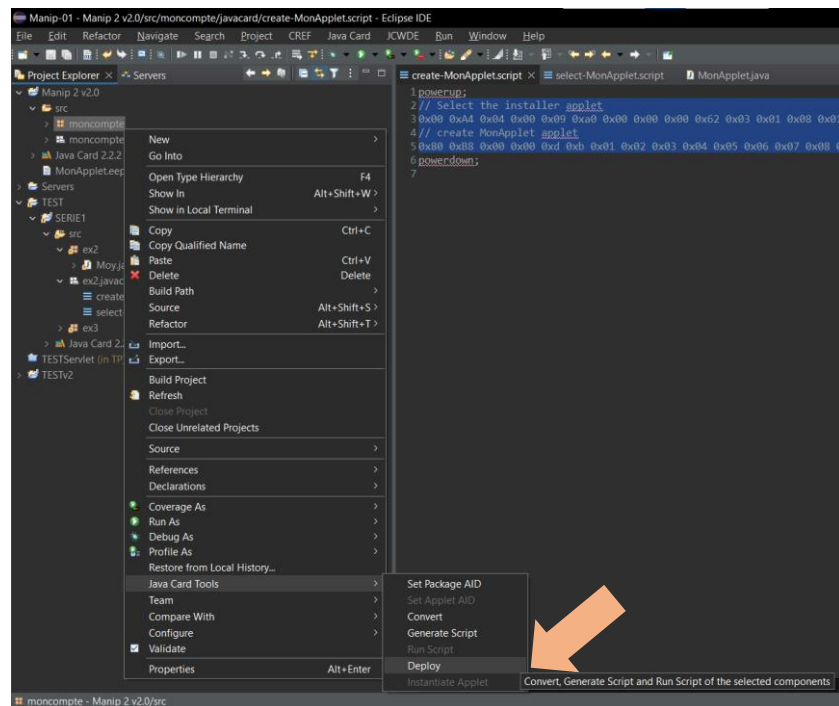


- On saisit « cmd » et on appuie sur entrer



Maintenant, on dispose d'une invite de commande située sous le dossier de notre projet

- **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

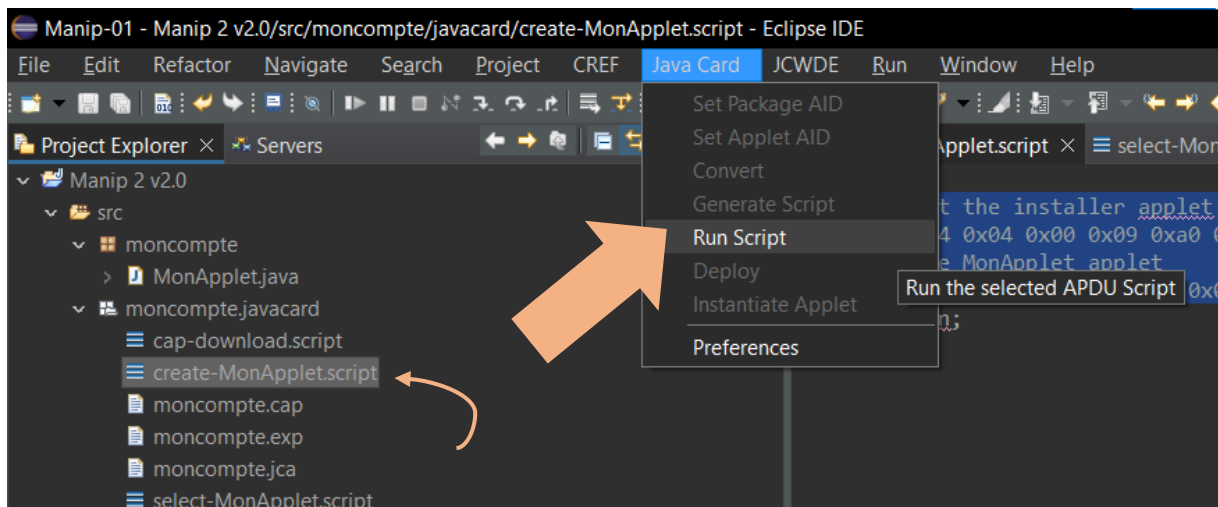
```
C:\Dhia\ING INFO 1\Semestre 1\Application logicielle d'entreprise\Manip-01\Manip 2 v2.0>cref -i MonApplet.eeprom
Java Card 2.2.2 C Reference Implementation Simulator (version 0.41)
32-bit Address Space implementation - with cryptography support
T=1 / T=CL Dual interface APDU protocol (ISO 7816-3)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

Memory configuration
  Type   Base   Size   Max Addr
  RAM    0x0     0x1000 0xffff
  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 effectuée.

- **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

```

Console x Problems Debug Shell Debug Output Browser Output Properties
<terminated> C:\Program Files\Java\jdk1.5.0_22\bin\javaw.exe (18 nov. 2023, 18:43:11) [pid: 21932]
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.
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x01
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
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

```

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 évidemment
- **On lance APDUTOOL** dans une autre invite de commande
- **On lance les commandes de sélection d'Applet**

```
Command Prompt - cref -p 9025 -i MonAppleteeprom
C:\Ohia\ING INFO 1\Semestre 1\Application logicielle d'entreprise\Manip-01\Manip 2 v2.0>cref -p 9025 -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   0xffff
ROM       0x2000   0xe000   0xffff
EEP       0x10020  0xffe0   0x1ffff

ROM Mask size =          0x064 =      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

Microsoft Windows [Version 10.0.19045.3448]
(c) Microsoft Corporation. All rights reserved.

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;
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x01
// Select the installer applet
0x00 0xA4 0x04 0x00 0x00 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 0x7F;
// create MonApplet applet
0xB0 0xB8 0x00 0x00 0xd 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x02 0x00 0x7F;
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
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, S
W2: 00
0xB0 0xB8 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 00, SW1: 90, SW2: 00
0xB0 0x03 0x0a 0x00 0x00 0x7F;
CLA: b0, INS: 03, P1: 0a, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 0a, SW1: 90, SW2: 00
0xB0 0x01 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 01, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 09, SW1: 90, SW2: 00
```

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



Compte Rendu Manip-03

Préparation Programmation d'une Application côté Client

Yassine El Kamel

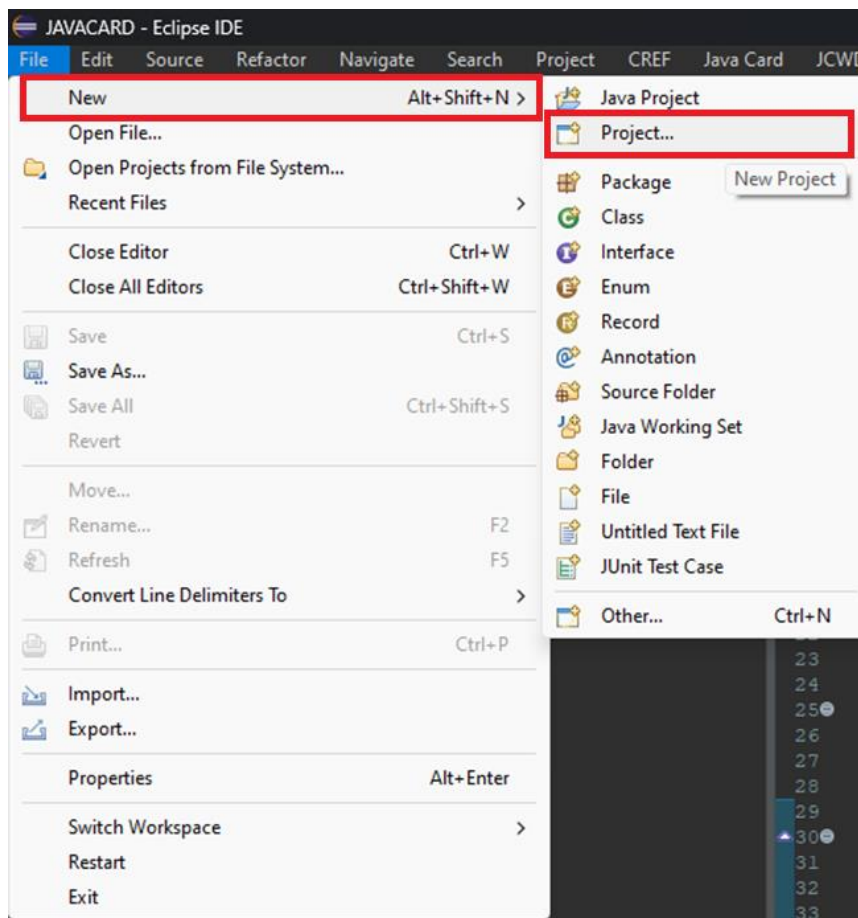
Dhia Ben Hamouda

ING INFO 1 TD1

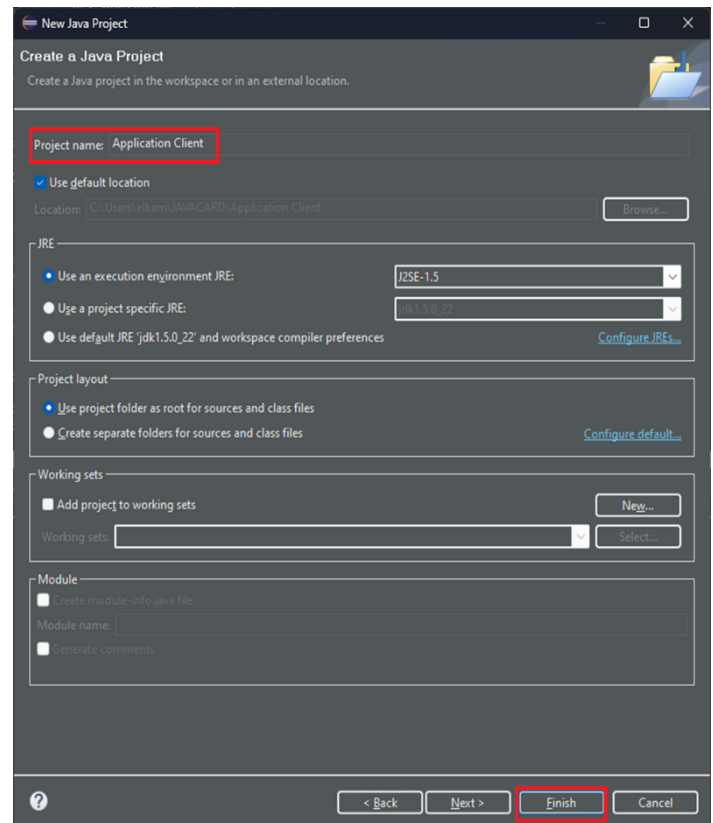
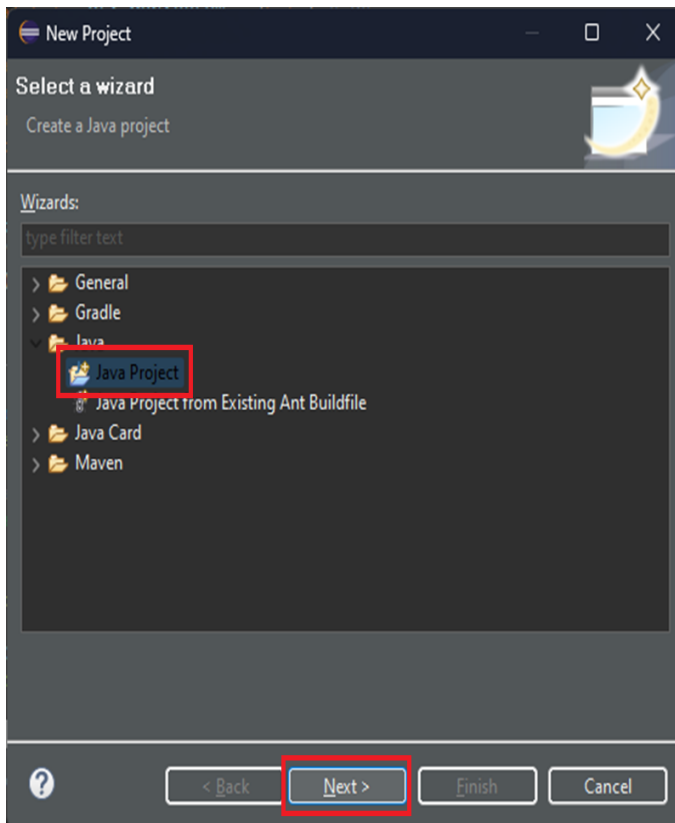
Création de l'application client sous Eclipse :

A. Création d'un nouveau projet :

Créons un nouveau projet. Pour cela, dans Eclipse, aller dans le menu File, faire New puis Project... :



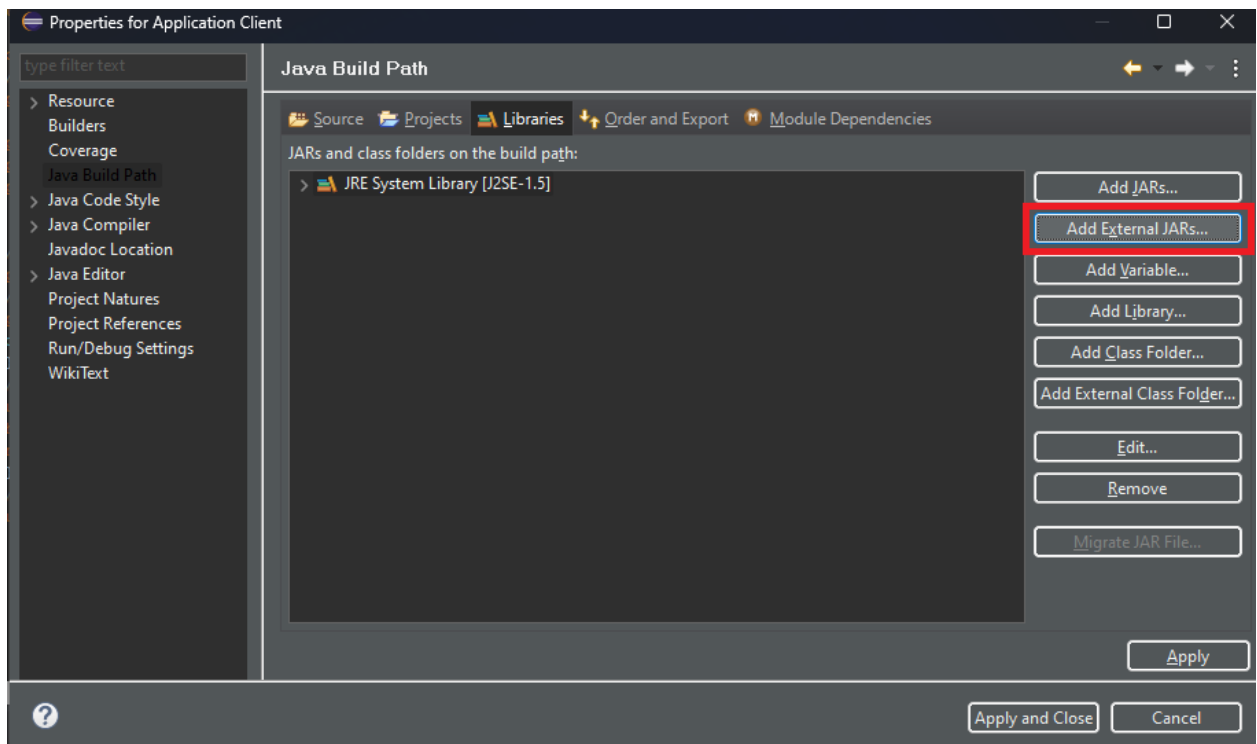
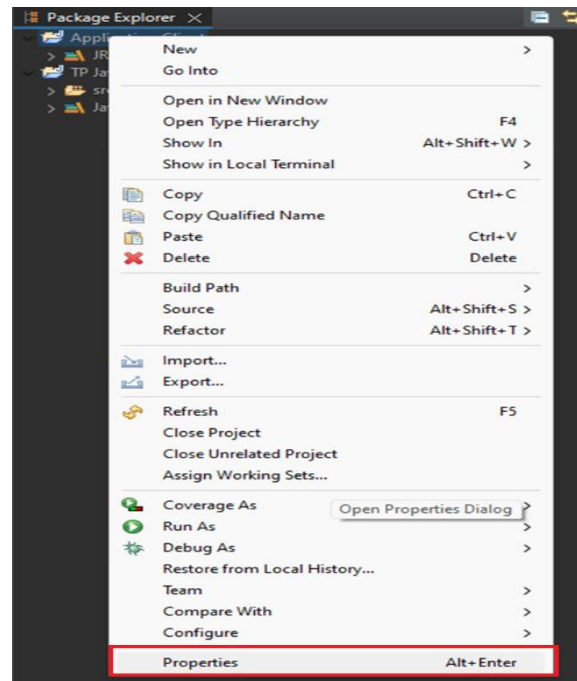
Choisir Java Project puis confirmer en cliquant sur Next. Donner un nom au nouveau projet (Application client par exemple) puis cliquer sur Finish :



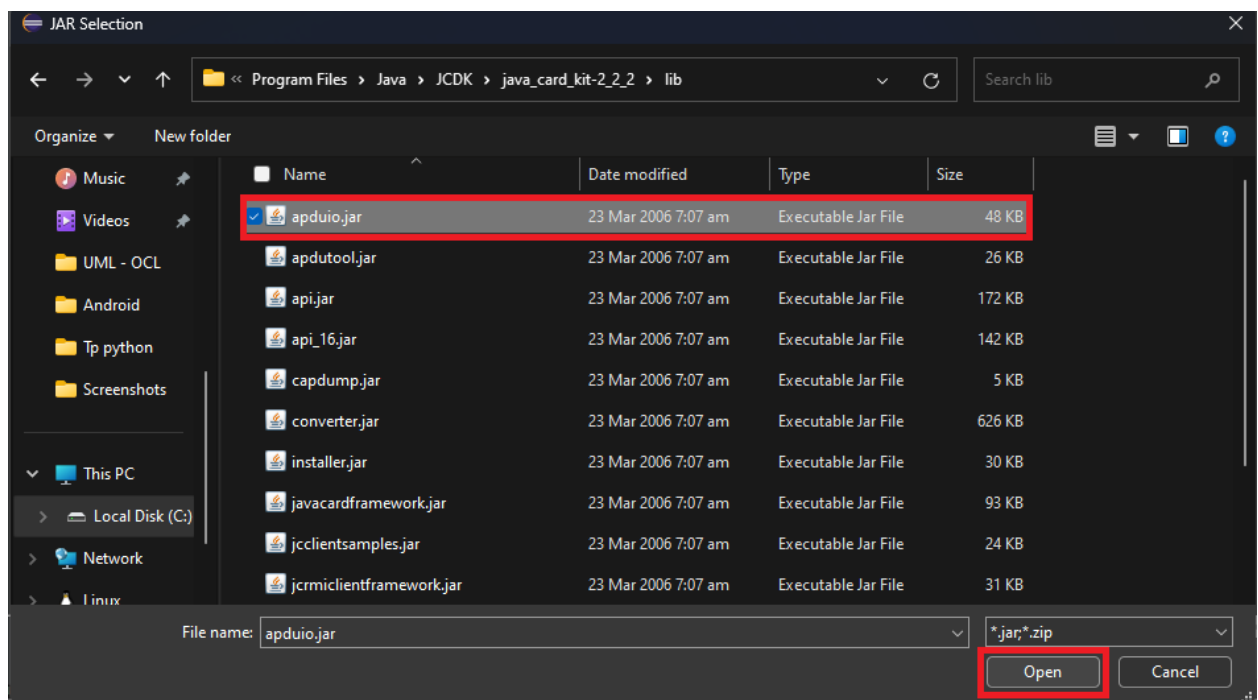
B. Ajout de la librairie « apduio » dans le classpath :

Afin de pouvoir utiliser les classes servant à communiquer avec notre Javacard, il faut ajouter la bibliothèque apduio.jar (présente dans le répertoire C:\JCDK\java_card_kit-2_2_2\lib).

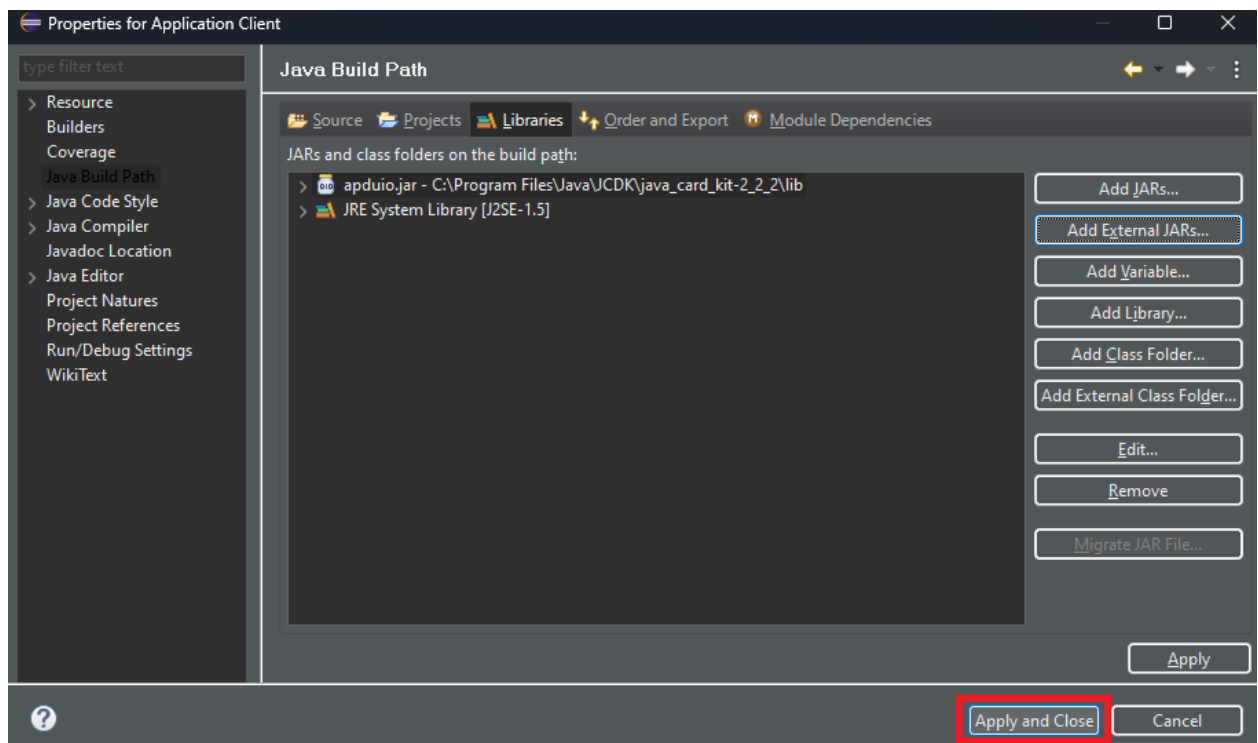
Dans la section Java Build Path, sélectionner l'onglet Librairies et cliquer sur le bouton Add External Jars :



On sélectionne le fichier apduio.jar qui se trouve dans le dossier %CATALINA_HOME%/lib

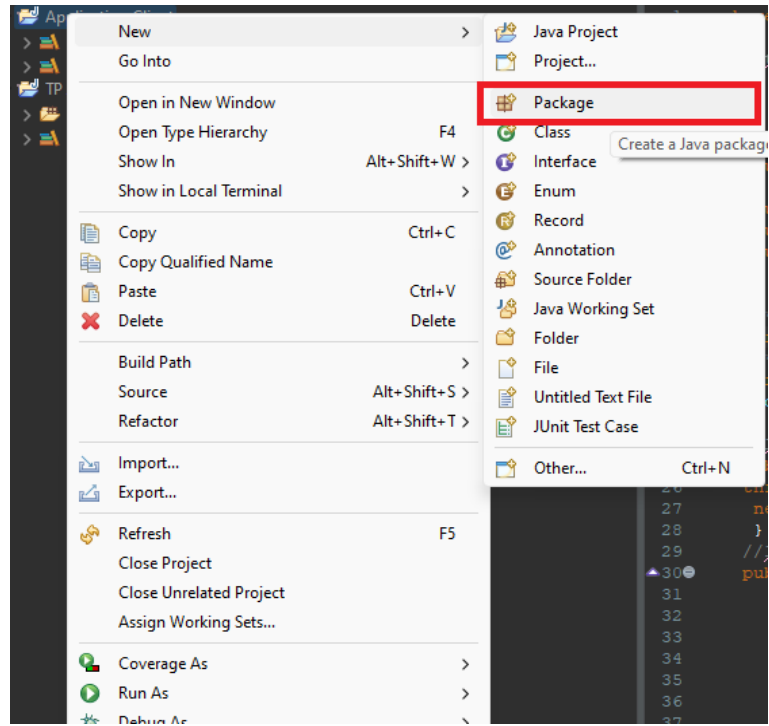


Puis appuyer sur le bouton « Apply and Close » :

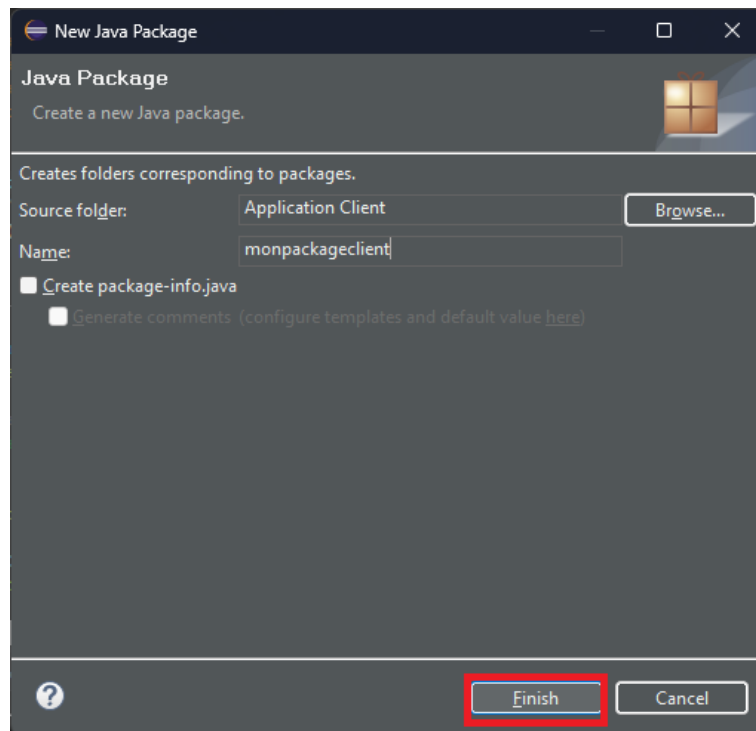


C. Création de la classe principale :

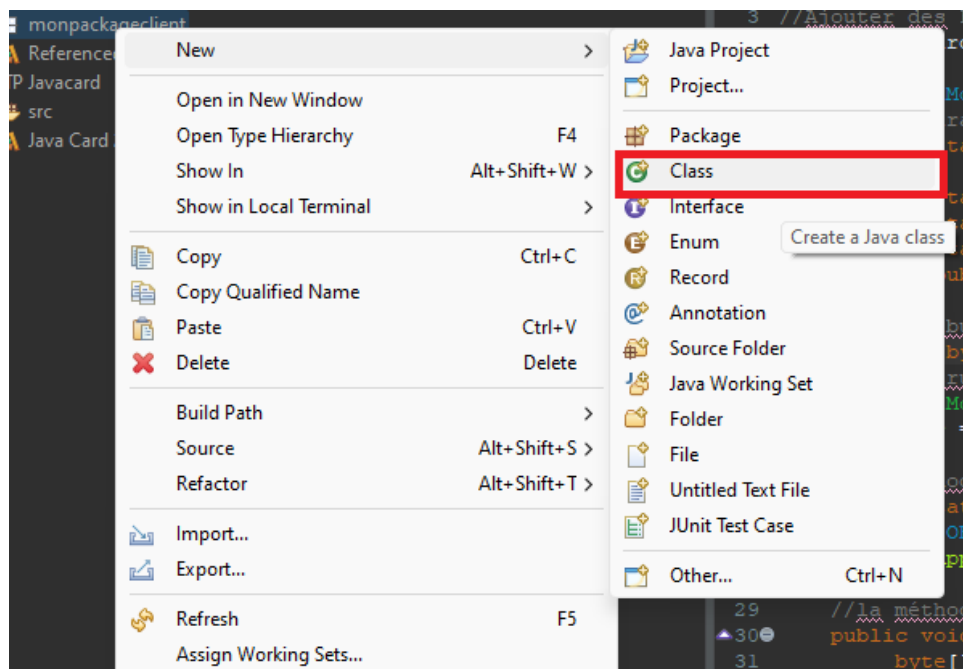
Tout d'abord, créons un nouveau package. Pour cela, faire un clic droit sur notre projet, puis New et Package :



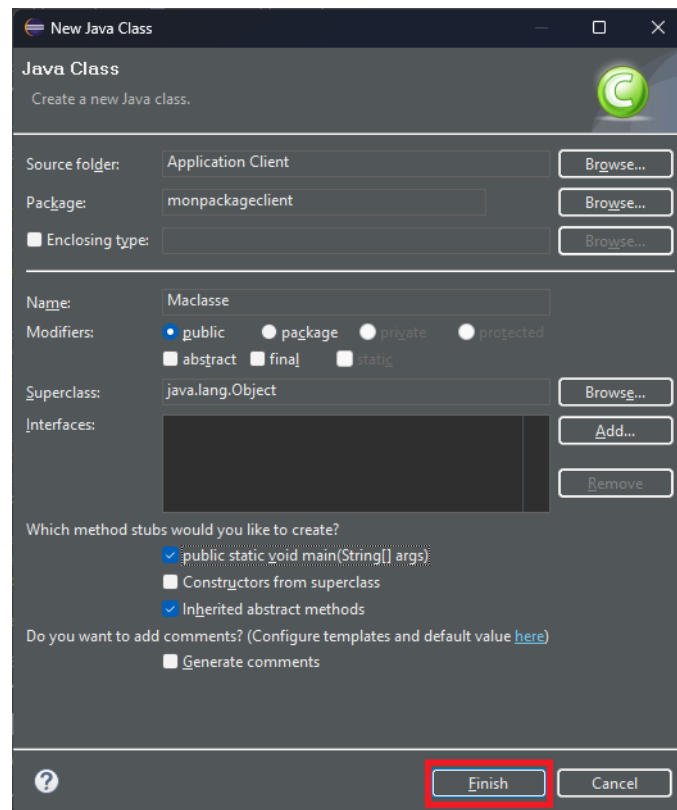
Donner un nom au package (monpackageclient par exemple), puis valider à l'aide du bouton Finish :



Créons maintenant la classe principale de notre application. Pour cela, faire un clic droit sur le package créé puis New et Class :



Donnons un nom à notre nouvelle classe (Maclasse par exemple), cocher la case public static void main puis cliquer sur Finish :



Les importations:

```
package monpackageclient;

import java.io.BufferedReader;
import java.io.BufferedOutputStream;
import java.io.IOException;
import java.net.Socket;
import com.sun.javacard.apduio.Apdu;
import com.sun.javacard.apduio.CadTlClient;
import com.sun.javacard.apduio.CadTransportException;

public class Maclasse {

    public static final byte CLA_MONAPPLET = (byte) 0xB0;

    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;

    public static void main(String[] args) {
```

L'application cliente se trouve sur le terminal qui communique avec le serveur (applet carte), on peut séparer l'écriture de notre application en plusieurs étapes :

Étape 1 - Connexion :

```
/* Connexion - Javacard */
CadT1Client cad;
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;
}

try {
    cad.powerUp();
} catch (Exception e2) {
    System.out.println("Erreur lors de l'envoi de la
commande Powerup 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;
}
```

Etape 2 Sélection :

```
/* Sélection de l'applet */

try {
    Adu apdu = new Adu();
    apdu.command[Adu.CLA] = 0x00;
    apdu.command[Adu.INS] = (byte) 0xA4;
    apdu.command[Adu.P1] = 0x04;
    apdu.command[Adu.P2] = 0x00;
    byte[] appletAID = { 0x01, 0x02, 0x03, 0x04, 0x05,
0x06, 0x07, 0x08, 0x09, 0x00, 0x00 };
}
```

```

        apdu.setDataIn(appletAID);
        cad.exchangeApu(apdu);
        if (apdu.getStatus() != 0x9000) {
            System.out.println("Erreur lors de la
sélection de l'applet");
            System.exit(1);
        }
    } catch (Exception e) {
        System.out.println("Erreur lors de la sélection de
l'applet");
        return;
    }
}

```

Etape 3 - Invocation des services implémentés :

```

/* Menu principal */
boolean fin = false;
while (!fin) {
    System.out.println();
    System.out.println("Application cliente
Javacard");
    System.out.println("-----");
    System.out.println();
    System.out.println("1 - Interroger le compteur");
    System.out.println("2 - Inrementer le compteur");
    System.out.println("3 - Decrementer le compteur");
    System.out.println("4 - Reinitialiser le
compteur");
    System.out.println("5 - Quitter");

    System.out.println();
    System.out.println("Votre choix ?");
    try {
        int choix = System.in.read();
        while (!(choix >= '1' && choix <= '5')) {
            choix = System.in.read();
        }

        Apdu apdu = new Apdu();
        apdu.command[Apdu.CLA] =
Maclasse.CLA_MONAPPLET;
        apdu.command[Apdu.P1] = 0x00;
        apdu.command[Apdu.P2] = 0x00;
        apdu.setLe(0x7f);
    }
}

```

```

        switch (choix) {
            case '1':
                try { apdu.command[Apdu.INS] =
Maclasse.INS_INTERROGER_COMPTEUR;
                    cad.exchangeApdu(apdu);
                    if (apdu.getStatus() != 0x9000) {
                        System.out.println("Erreur :
status word different de 0x9000");
                    } else {
                        System.out.println("Valeur du
compteur : " + apdu.dataOut[0]);
                    }
                } catch (Exception e) {
                    System.out.println("Erreur :
status word different de 0x9000");
                }
                break;

            case '2':
                try { apdu.command[Apdu.INS] =
Maclasse.INS_INCREMENTER_COMPTEUR;
                    cad.exchangeApdu(apdu);
                    if (apdu.getStatus() != 0x9000) {
                        System.out.println("Erreur :
status word different de 0x9000");
                    } else {
                        System.out.println("OK");
                    }
                } catch (Exception e) {
                    System.out.println("Erreur :
status word different de 0x9000");
                }
                break;

            case '3':
                try { apdu.command[Apdu.INS] =
Maclasse.INS_DECREMENTER_COMPTEUR;
                    cad.exchangeApdu(apdu);
                    if (apdu.getStatus() != 0x9000) {
                        System.out.println("Erreur :
status word different de 0x9000");
                    } else {
                        System.out.println("OK");
                    }
                }
                catch (Exception e) {
                    System.out.println("Erreur :
status word different de 0x9000");
                }
                break;
        }

```

```

        case '4':
            try{apdu.command[Apdu.INS] =
Maclasse.INS_INITIALISER_COMPTEUR;
                byte[] donnees = new byte[1];
                donnees[0] = 0;
                apdu.setDataIn(donnees);
                cad.exchangeApdu(apdu);
                if (apdu.getStatus() != 0x9000) {
                    System.out.println("Erreur :
status word different de 0x9000");
                } else {
                    System.out.println("OK");
                }
            } catch (Exception e) {
                System.out.println("Erreur :
status word different de 0x9000");
            }

            break;

        case '5':
            fin = true;
            break;

    }
} catch (Exception e){
    System.out.println("Erreur reading input");
}
}

```

Etape 4 - Mise hors tension :

La déconnexion de la Javacard se fera via la méthode powerDown() de la classe CadT1Client :

```

    /* Mise hors tension de la carte */
    try {
        cad.powerDown();
    } catch (Exception e) {
        System.out.println("Erreur lors de l'envoi de la
commande Powerdown a la Javacard");
        return;
    }
}
}

```

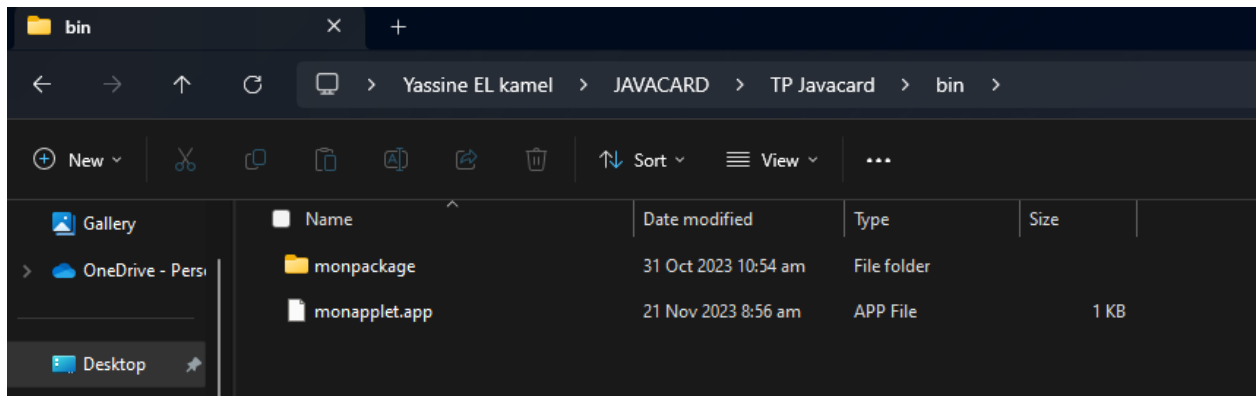
D. Utilisation de l'application cliente avec un simulateur – JCWDE :

Afin de pouvoir lancer le simulateur de notre applet en ligne de commande, nous allons créer un fichier "de configuration". Il permet de lister les applets Javacard à installer pour la simulation et de spécifier leurs AID respectifs.

- Créons notre fichier que nous appellerons "monapplet.app" (répertoire parent du package contenant le fichier class de l'applet Java Card « .\workspace\TP Javacard\bin ») . Il contiendra la ligne suivante :

```
monpackage.MonApplet 0x01:0x02:0x03:0x04:0x05:0x06:0x07:0x08:0x09:0x00:0x00
```

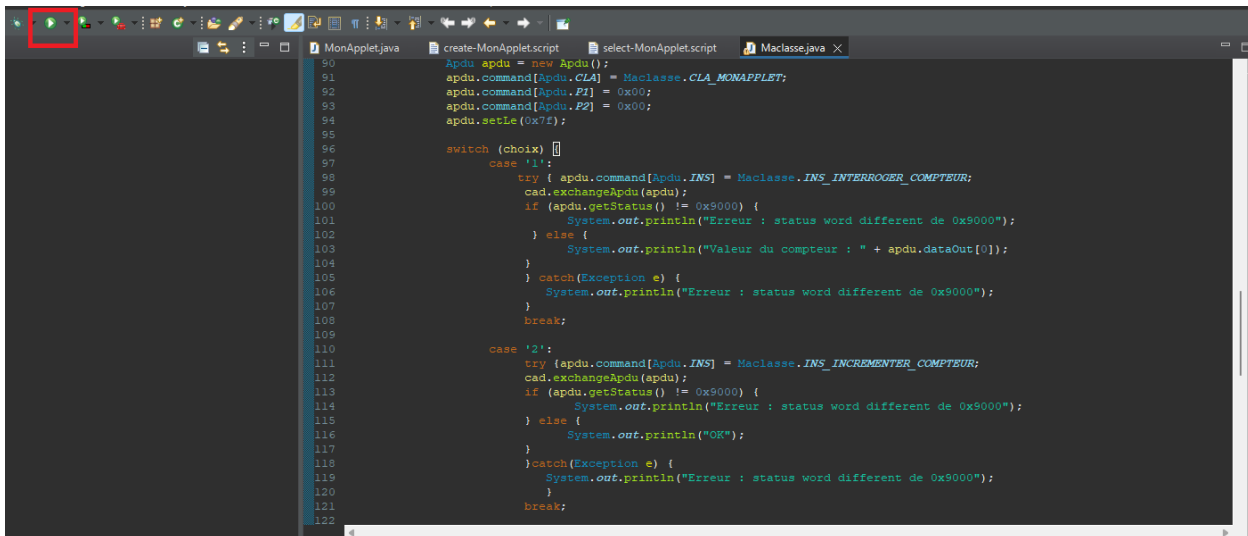
Le fichier est créé et mis sous le répertoire : « C:\Users\elkam\JAVACARD\TP Javacard\bin »



- Nous pouvons maintenant lancer notre simulateur, en prenant soin de nous placer dans le bon répertoire (répertoire parent du package contenant le fichier class de l'applet « C:\Users\elkam\JAVACARD\TP Javacard\bin » dans notre exemple) :

```
C:\Users\elkam\JAVACARD\TP Javacard\bin>jcwde monapplet.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 Adu's on TCP/IP port 9,025.
```

- Maintenant que notre simulateur est lancé, lançons notre application cliente en exécutant la classe MaClasse :

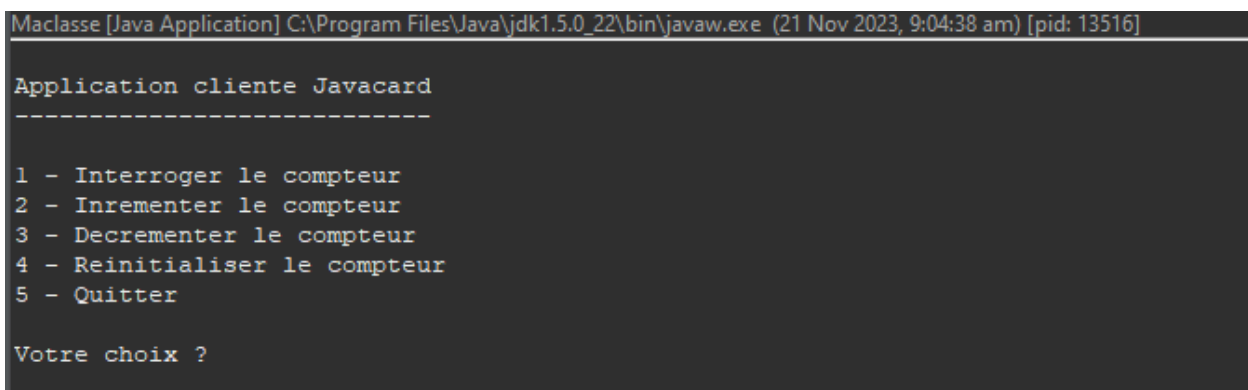


```

90  Adu adu = new Adu();
91  adu.command[Adu.CLA] = Maclasse.CLA_MONAPLET;
92  adu.command[Adu.PI] = 0x00;
93  adu.command[Adu.P2] = 0x00;
94  adu.setLe(0x7f);
95
96  switch (choix) {
97      case '1':
98          try { adu.command[Adu.INS] = Maclasse.INS_INTERROGER_COMPTEUR;
99              cad.exchangeAdu(adu);
100              if (adu.getStatus() != 0x9000) {
101                  System.out.println("Erreur : status word different de 0x9000");
102              } else {
103                  System.out.println("Valeur du compteur : " + adu.dataOut[0]);
104              }
105              } catch (Exception e) {
106                  System.out.println("Erreur : status word different de 0x9000");
107              }
108              break;
109
110      case '2':
111          try { adu.command[Adu.INS] = Maclasse.INS_INCREMENTER_COMPTEUR;
112              cad.exchangeAdu(adu);
113              if (adu.getStatus() != 0x9000) {
114                  System.out.println("Erreur : status word different de 0x9000");
115              } else {
116                  System.out.println("OK");
117              }
118              } catch (Exception e) {
119                  System.out.println("Erreur : status word different de 0x9000");
120              }
121              break;
122

```

Résultat de l'exécution :



```

Maclasse [Java Application] C:\Program Files\Java\jdk1.5.0_22\bin\javaw.exe (21 Nov 2023, 9:04:38 am) [pid: 13516]

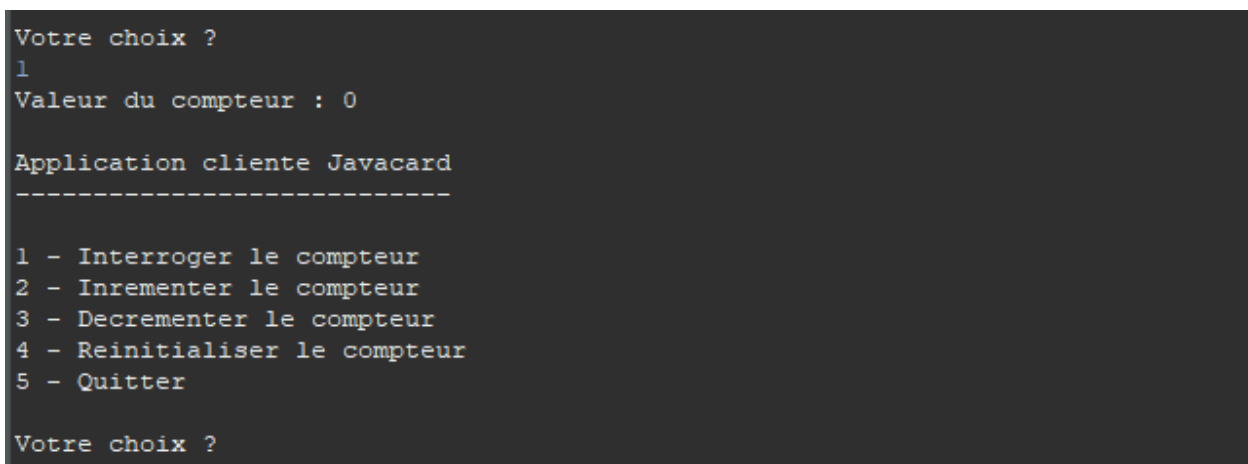
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?

```

- Commençons par interroger le compteur : tapons 1 puis validons :



```

Votre choix ?
1
Valeur du compteur : 0

Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?

```


Le compteur est bien à 0.

Incrémentons maintenant le compteur 3 fois puis décrémentation-le une fois. Lorsque nous réinterrogeons la carte, le compteur vaut 2 donc tout va bien :

- Incrémentation 3 fois :

```
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?
2
OK

Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?
2
OK

Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?
2
OK
```

- Décrémentation une seule fois :

```
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?
3
OK
```

- Interrogant maintenant le compteur pour voir le résultat :

```
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?

1
Valeur du compteur : 2
```

```
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?

1
Valeur du compteur : 0
```

La valeur du compteur est bien 2 donc c bon.

▪ Quittons maintenant notre application cliente (commande 5). Nous pouvons voir que le simulateur se termine automatiquement à la réception de la commande "powerdown" :

```
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Inrementer le compteur
3 - Decrementer le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?

5
```

Résultat dans le cmd: "jcwde exiting on receipt of power down command."

```
Valeur du compteur : 2

Application cliente Javacard
-----

1 - Interroger le compteur
2 - Incrémenter le compteur
3 - Décrémenter le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?
5
```

```
Command Prompt
C:\Users\elkam\JAVACARD\TP Javacard\bin>jcwde monapplet.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 Aduis on TCP/IP port 9,025.
jcwde exiting on receipt of power down command.
```

Remarque:

Le simulateur JCWDE ne permet pas de conserver l'état de la carte. Si nous le relançons, le compteur sera à 0 et non à 2.

Valeur de compteur = 0 après la relance du simulateur :

```
Application cliente Javacard
-----

1 - Interroger le compteur
2 - Incrémenter le compteur
3 - Décrémenter le compteur
4 - Reinitialiser le compteur
5 - Quitter

Votre choix ?
1
Valeur du compteur : 0
```



Rapport Projet Final

Mise en place d'une plateforme de Distributeur d'argent automatique

Yassine El Kamel

Dhia Ben Hamouda

ING INFO 1 TD1

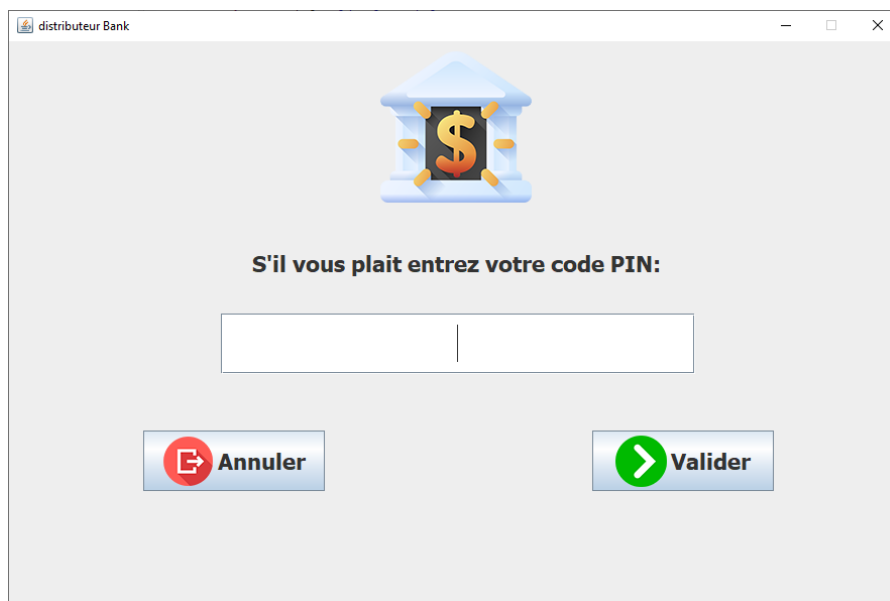
Interface Graphique

A. Vérification du Code PIN

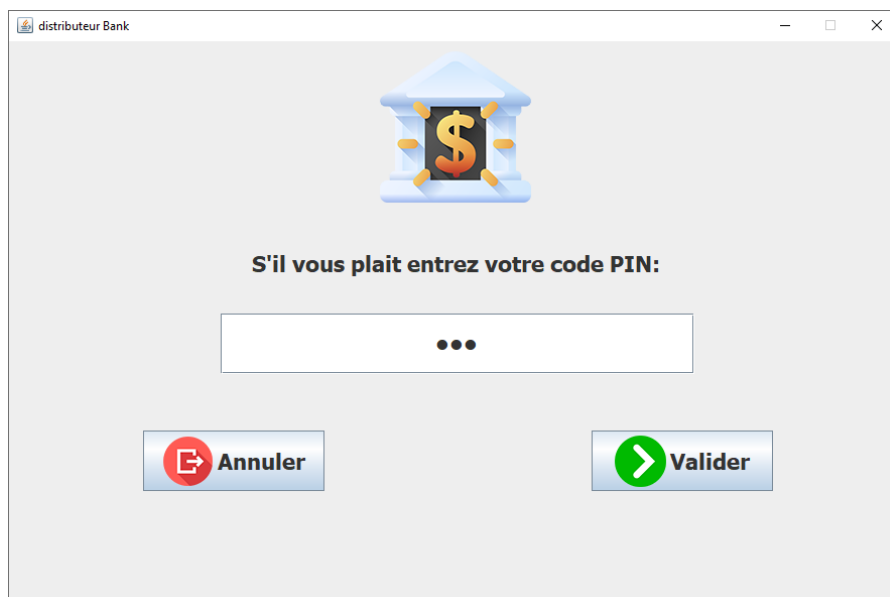
L'interface de vérification du Code PIN est la première interface qui s'affiche lors de l'exécution.

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)



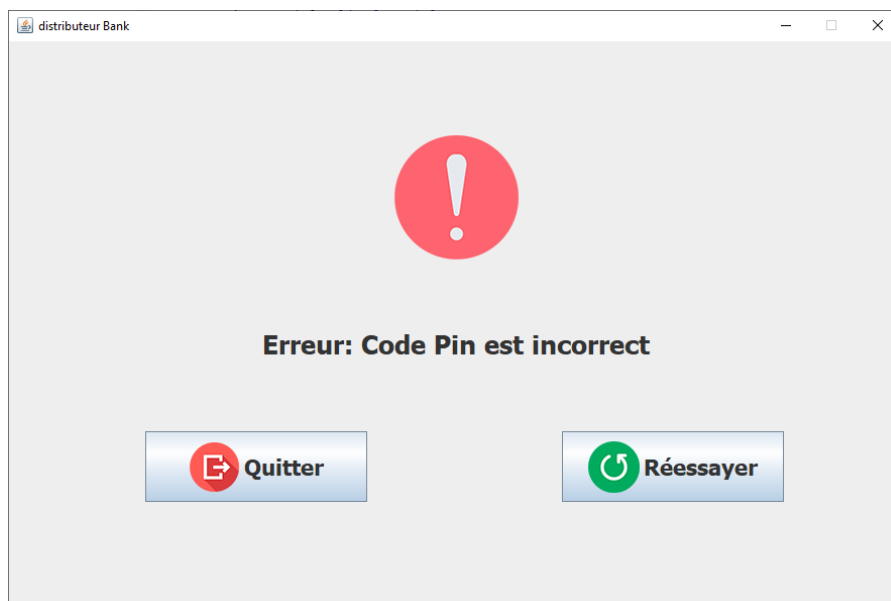
- **3 Cas se présentent :**
 - a) **Le code PIN est Correct :**

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

- b) **Le Code PIN est erroné :**

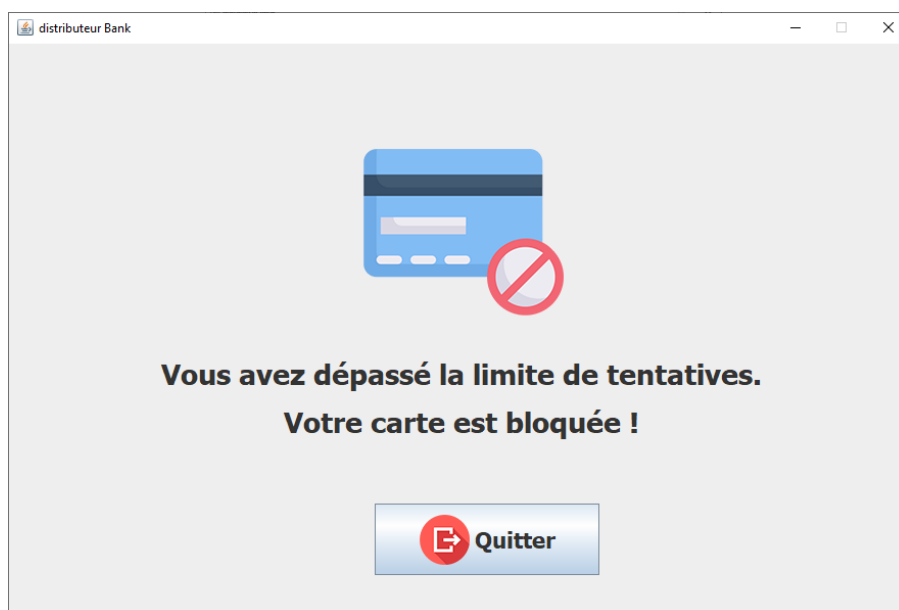


Vous êtes présenté avec un message d'erreur. Aussi le nombre d'essais permis est décrémenté par 1.



c) Le nombre maximal de tentatives est atteint :

La carte est alors bloquée. De nouvelles tentatives de validation ne sont pas permises. 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



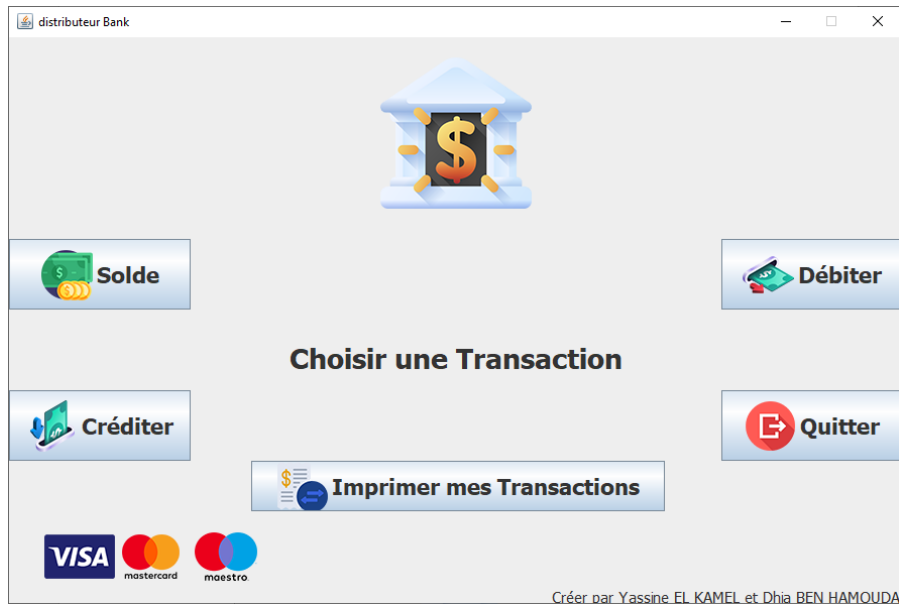
- **Bouton Quitter** : Ferme l'application et envoie la commande « powerdown » à la carte



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 Adu's on TCP/IP port 93025.
jcwde exiting on receipt of power down command.
```

B. Menu Solde



C. Le Menu Solde affiche le solde de votre compte.

- Pour afficher le solde, deux styles différents peuvent se présenter :

a) Solde de valeur supérieur ou égal à 0 :



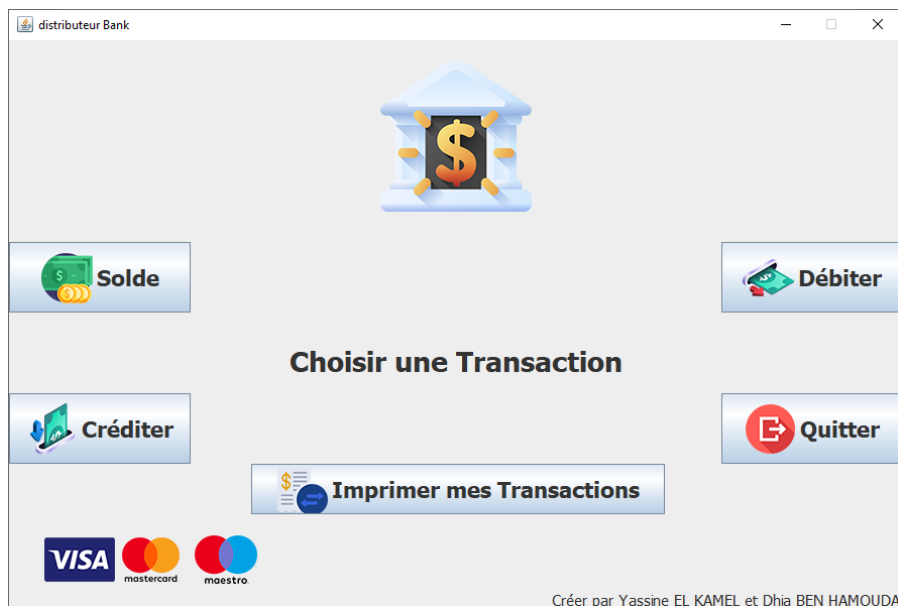
Le solde est affiché avec un style par défaut

b) Solde de valeur inférieur à 0 :

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

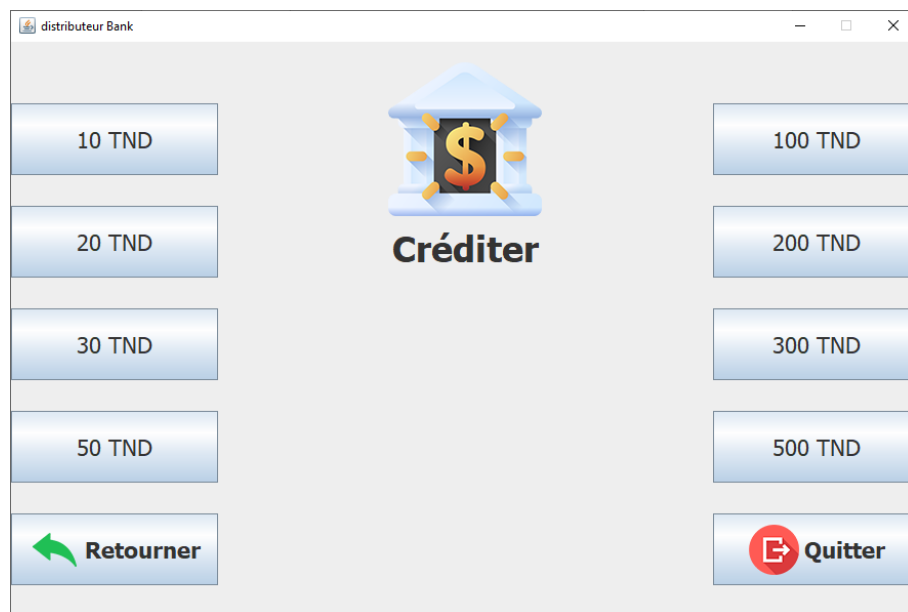


D.Menu Créditer

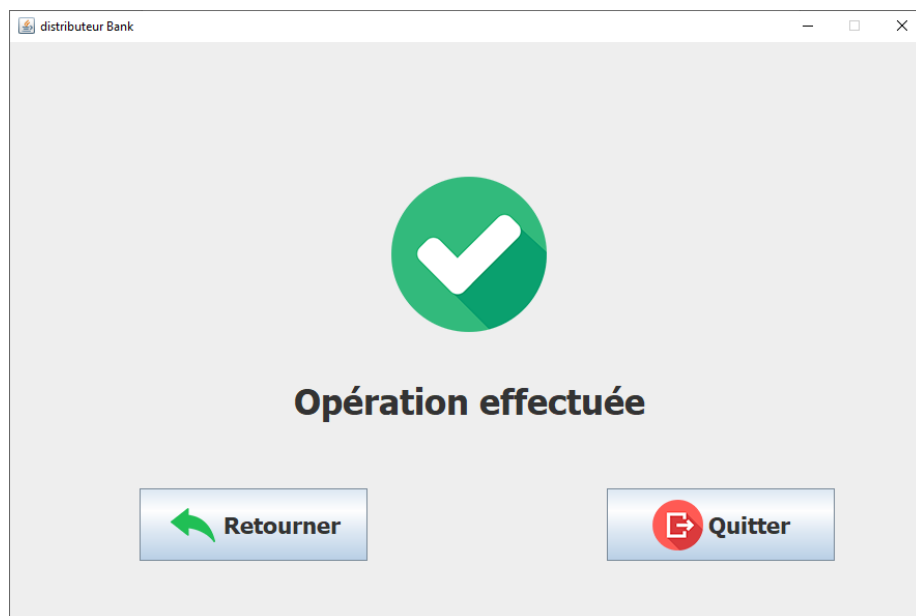


Ce menu sert à incrémenter le solde. Ceci implémente la fonctionnalité d'insérer de l'espèce dans le distributeur afin d'incrémenter le solde.

Ceci est le menu créditer



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

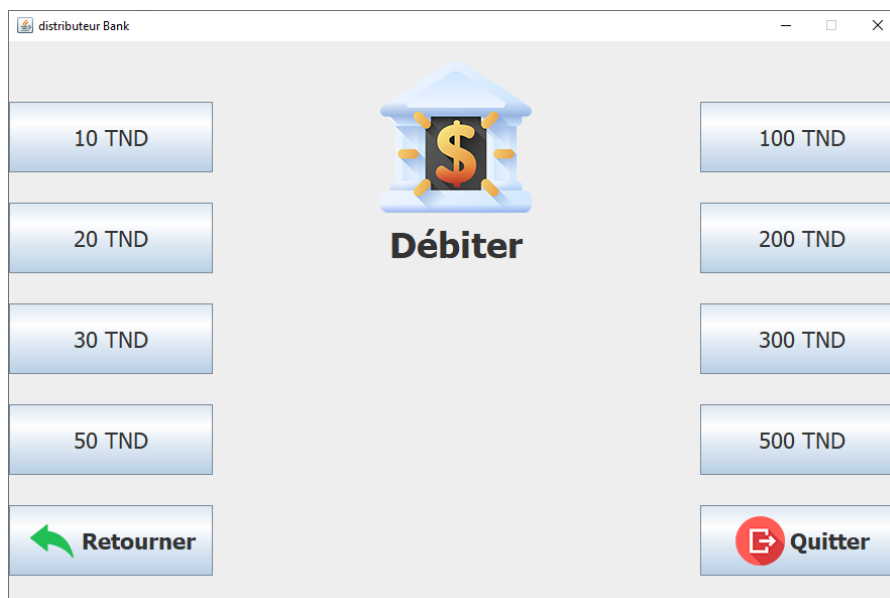


E. Menu Débiter

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



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



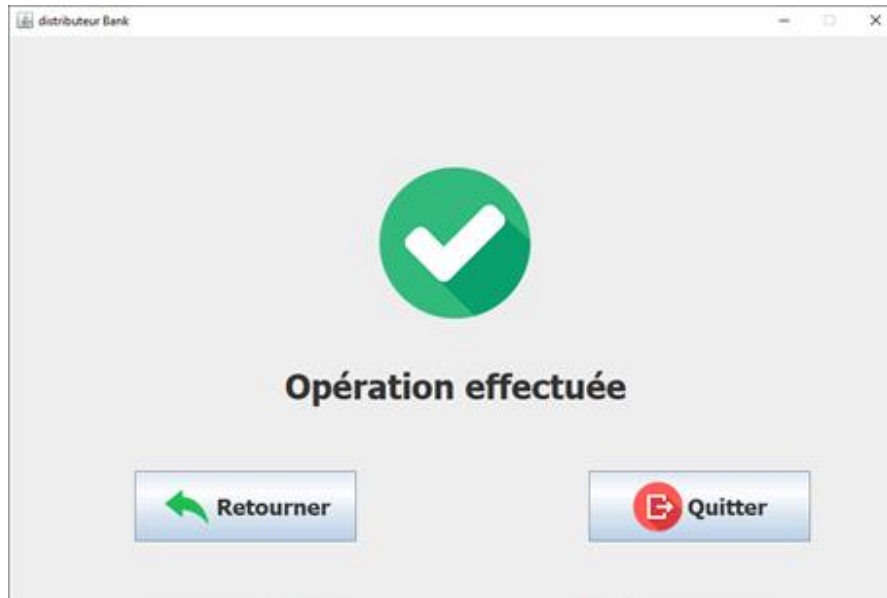
3 Cas se présentent :

a) Opération avec succès :

Votre solde est supérieur au montant choisi.

Vous avez assez de solde à débiter.

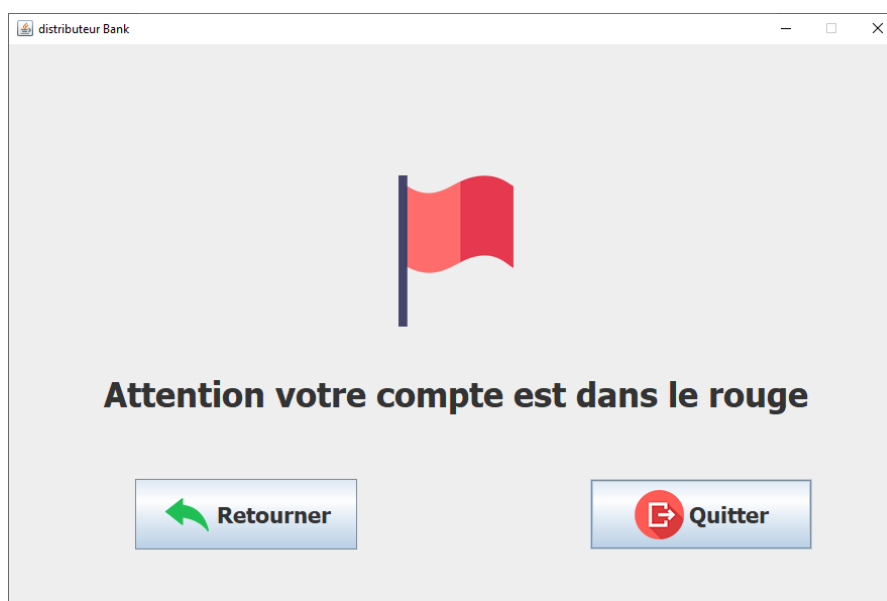
L'opération se déroule normalement.



b) Votre compte est dans le rouge :

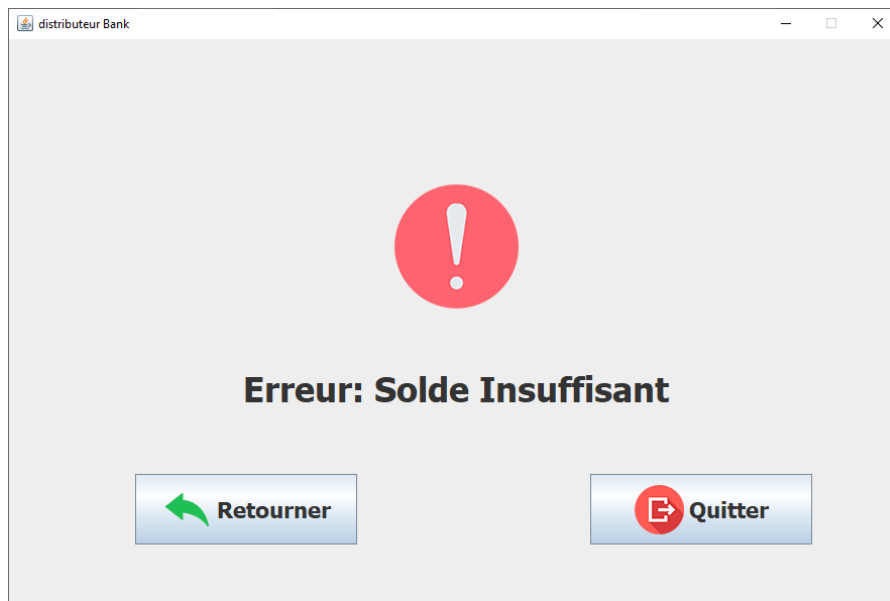
Votre solde est insuffisant, mais votre retrait d'argent ne dépasse pas le plafond de rouge permis.

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

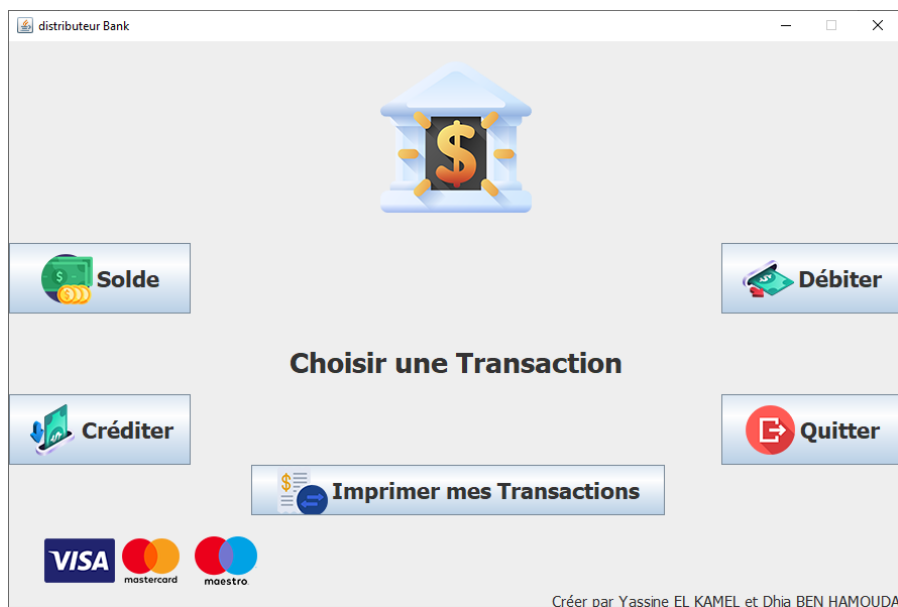


c) Solde Insuffisant :

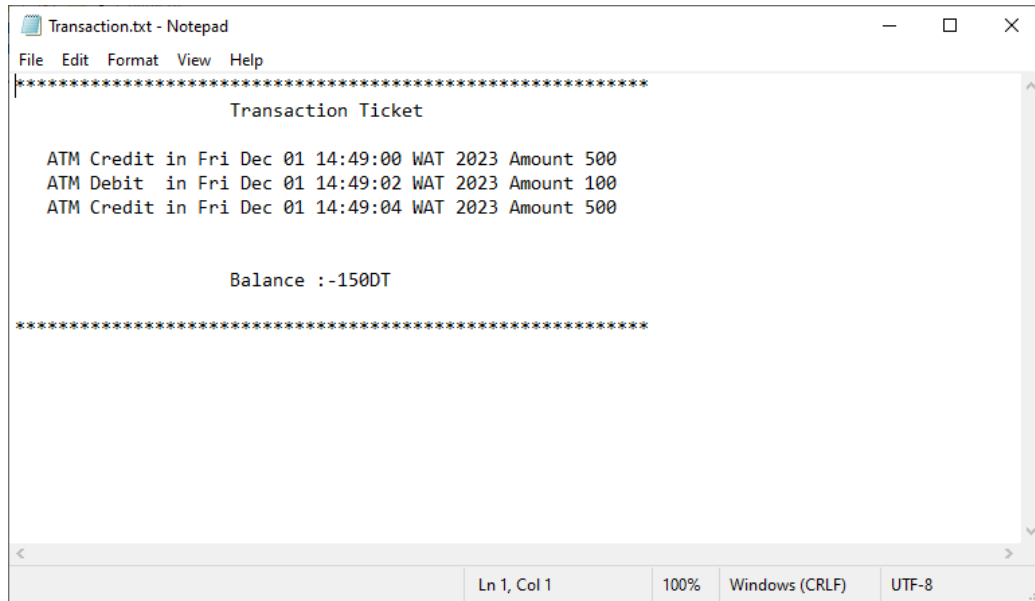
Votre est insuffisant et dépasse le plafond de rouge permis. L'argent n'est pas débité. Un message d'erreur s'affiche.



F. Menu Imprimer mes Transactions



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



```
Transaction.txt - Notepad
File Edit Format View Help
*****
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
*****

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

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

A.Implémentation du ATM

Différentes diverses fonctionnalités sont implémentées dans cette Applet

a) Hachage du Code PIN

Afin de stocker d'une manière sûre le code PIN, on ne doit pas le garder sous sa forme originale, 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 (incrément, 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 ;

B. 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 Echoué
private final static short SW_VERIFICATION_FAILED = 0x6300;

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

// signal that the balance exceeds 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;

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

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

// Longueur 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 initialise

```
private Bank(byte[] bArray,int i,int j) {
    balance = MAX_ROUGE;
    transition_buffer = JCSysm.makeTransientByteArray(
        (short)TRANSIENT_BUFFER_SIZE,
        JCSysm.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 Adu Msg(byte ins, byte lc, byte[] data,byte le) throws IOException,
CadTransportException{
    apdu = new Adu();
    apdu.command[Adu.CLA] = (byte) 0xB0;
    apdu.command[Adu.P1] = 0x00;
    apdu.command[Adu.P2] = 0x00;
    apdu.command[Adu.INS] = ins;
    //apdu.setLe(0x7f);
    apdu.setLe(le);
    if (data!=null)
        apdu.setDataIn(data);
    cad.exchangeAdu(apdu);
    return apdu;
}

void updateBalance() throws IOException, CadTransportException {
    Adu apdu = null;
    // transition_buffer Size defined in Bank.java (JavaCard Applet)
    byte size = 127;
    // Data Array transferred 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);
}
}

```