

Institut Supérieur d'Informatique et de Mathématiques de Monastir



Niveau :

**Cycle d'Ingénieur
en informatique**

Groupe :

TD1

Réaliser par :

Nour elhouda salem

Ahmed Chebbi

Enseignant : Mr Ramzi Mahmoudi



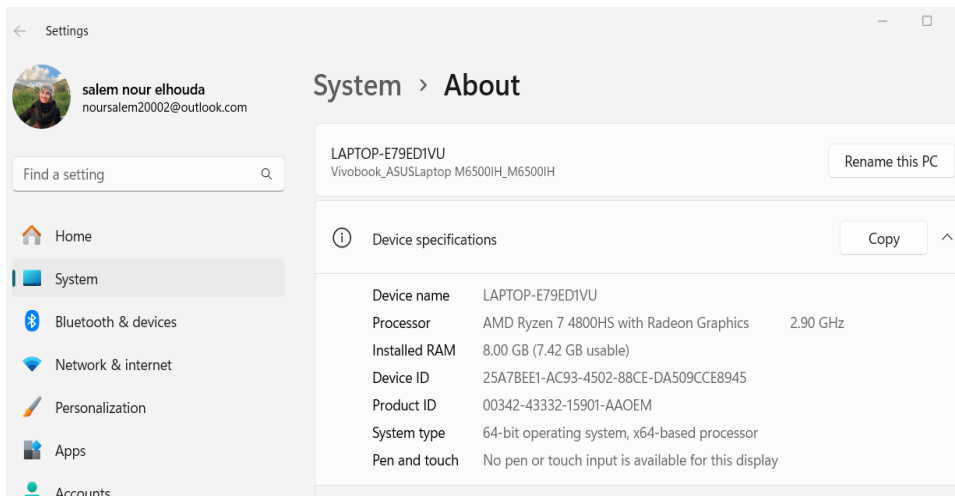
Table des matières

Table des matières.....	3
CHAPITRE 1. ENVIRONNEMENT DE DEVELOPPEMENT JAVACARD	4
I.1 Outils logiciels nécessaires.....	Error! Bookmark not defined.
I.1.1 Identification de sys d'exploitation:	Error! Bookmark not defined.
I.1.2 Telechargement de java card kit 2.2.2:	Error! Bookmark not defined.
I.1.3 Telechargement de java SE:.....	5
I.1.4 Telechargement de ECLIPSE IDE:.....	7
I.1.5 Telechargement de plugins d'integration:.....	8
I.2 Instruction d'installation:	Error! Bookmark not defined.
I.2.1 Installation d'ECLIPSE:.....	8
I.2.2 Installation du Java Card Development Kit 2.2.2 :	Error! Bookmark not defined.
I.2.3 Mise à jour des plugins Eclipse-JCDE:.....	Error! Bookmark not defined.
I.2.4 Conclusion :	18
CHAPITRE II. DEVELOPPEMENT D'UNE APPLICATION COTE SERVEUR	19
II.1.1 Création d'un nouveau Projet JavaCard :	19
II.1.2 Création d'une applet Javacard:	Error! Bookmark not defined.
II.2 Codage de notre applet :	22
II.3. Outils de Simulation :	24
II.3.1 JCWDE simulateur sans conservation d'etat:	24
II.3.2 CREF simulateur avec conservation d'etat:.....	28
CHAPITRE III. PROGRAMMATION D'UNE APPLICATION COTE CLIENT.....	31
III.1 Création de l'application client sous Eclipse :	31
III.1.1 Création d'un nouveau projet Java :	31
III.1.2 Ajout de la librairie « apduio » dans le classpath :	32
III.1.3 Création de la classe principale :	33
III.2 Utilisation de l'application cliente avec un simulateur – JCWDE :	38
CHAPITRE IV. REALISATION DU MINI-PROJET	41
IV.1 Interface Graphique (Partie Client) :	41
IV.1.1 WindowBuilder Introduction :	41
IV.1.2 Installation de WindowBuilder :	42
IV.1.3 Comment utiliser WindowBuilder :	44
IV.1.4 Les interfaces de notre application Client :	49
IV.1.5 Implémentation des méthodes nécessaires :	57
IV.2 Partie Serveur (Back-End) :	59
IV.2.1 Déclaration des variables et des constantes :	59
IV.2.2 Implémentations des méthodes nécessaires :	59

Chapitre 1. Environnement de développement JavaCard

I.1 Outils logiciels nécessaires :

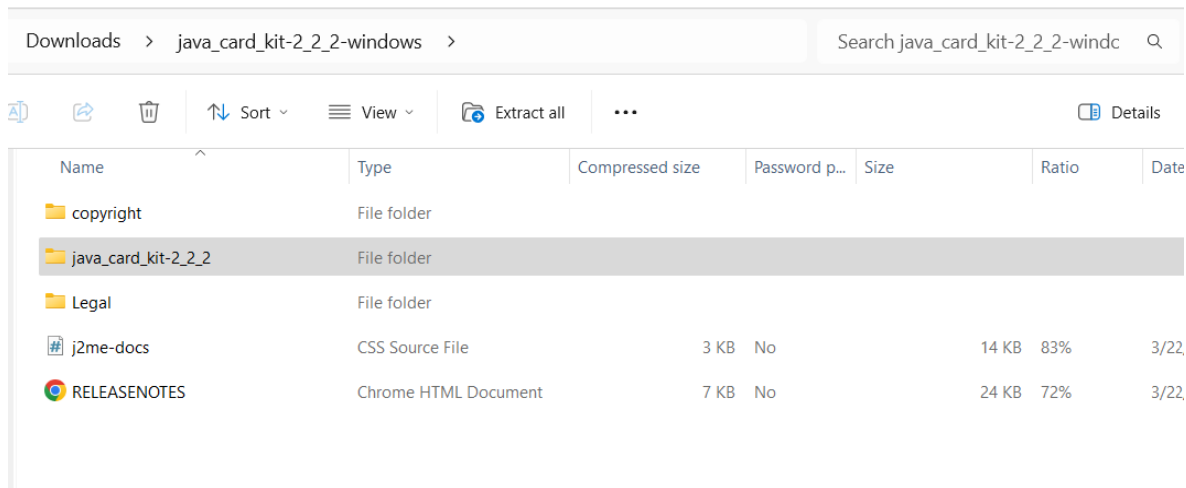
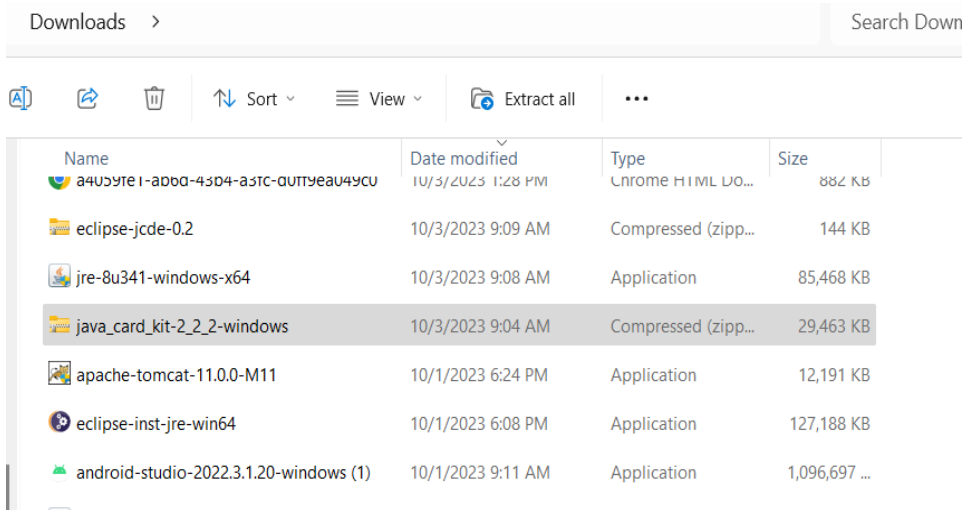
I.1.1. Identifiez votre système d'exploitation ainsi que son type



I.1.2 Telechargement Java Card Development Kit 2.2.2 :












Le Java Card Development Kit 2.2.2 : archive [java_card_kit-2_2_2-windows.zip](http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javame-419430.html) à partir du site d'Oracle :

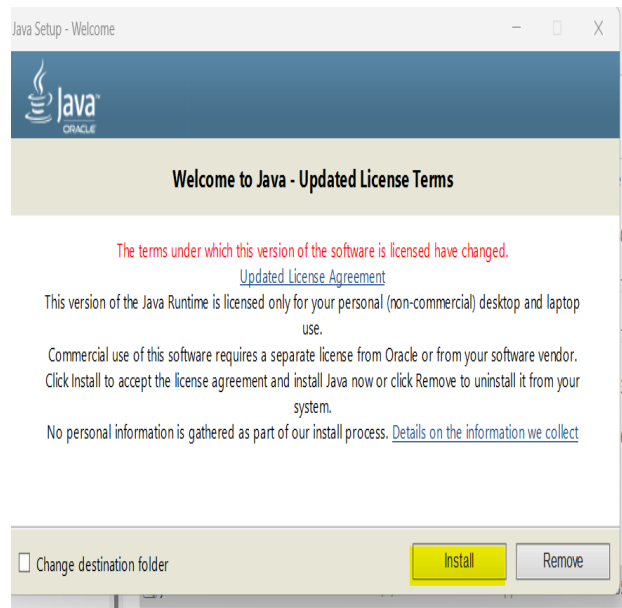
<http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javame-419430.html>



I.1.3 Téléchargement Java SE:

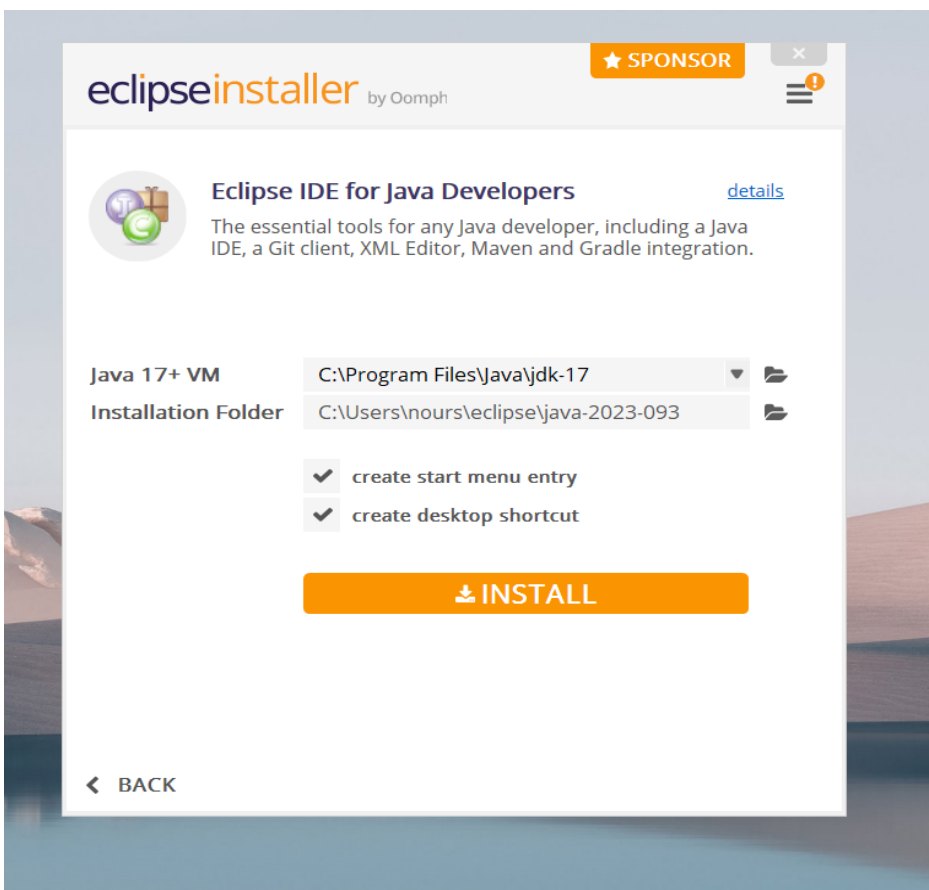
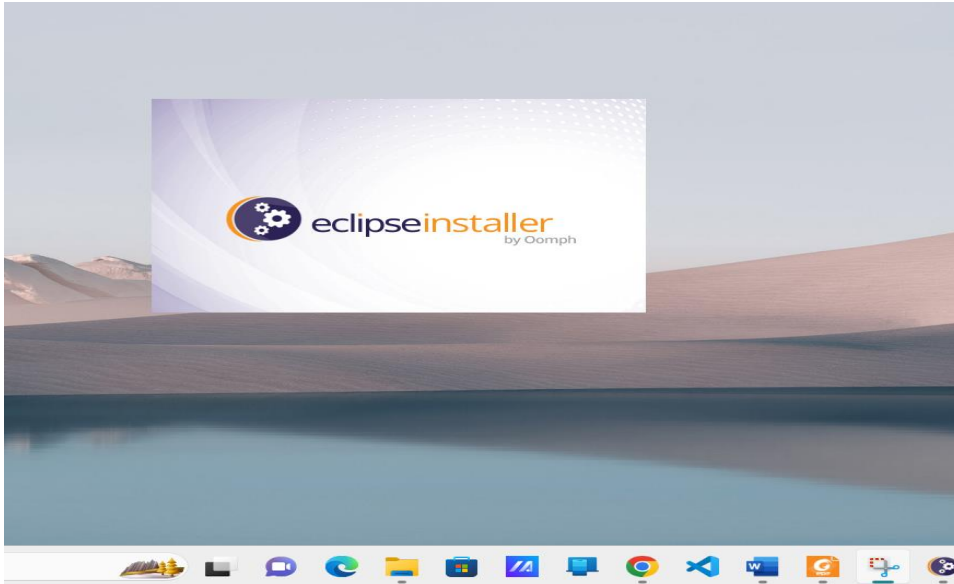
Télécharger un environnement standard édition de Java (Java SE) qui contient non seulement un **Java Runtime Environnement (JRE)** pour pouvoir tourner des applications java mais également un **Java Développement Kit** pour pouvoir compiler vos codes sources. Nous proposons de télécharger Java SE 7u. **link :** <http://www.oracle.com/>

 R-4.3.1-win	10/7/2023 8:54 AM	Application	80,173 KB
 eclipse-inst-jre-win64 (1)	10/4/2023 10:06 PM	Application	127,188 KB
 jdk-17.0.8_windows-x64_bin	10/4/2023 9:20 PM	Application	157,169 KB
 apache-tomcat-10.1.13	10/4/2023 9:06 PM	Application	13,215 KB
 jdk-8u202-windows-x64	10/4/2023 8:56 PM	Application	216,653 KB
 a4059fe1-ab6d-43b4-a3fc-d0ff9ea049c0	10/3/2023 1:28 PM	Chrome HTML Do...	882 KB
 eclipse-jcde-0.2	10/3/2023 9:09 AM	Compressed (zipp...	144 KB
 jre-8u341-windows-x64	10/3/2023 9:08 AM	Application	85,468 KB
 java_card_kit-2_2_2-windows	10/3/2023 9:04 AM	Compressed (zipp...	29,463 KB
 apache-tomcat-11.0.0-M11	10/1/2023 6:24 PM	Application	12,191 KB
 eclipse-inst-jre-win64	10/1/2023 6:08 PM	Application	127,188 KB



I.1.4 Téléchargement ECLIPSE IDE:







Suivant le type de votre système, télécharger un environnement de développement java pour pouvoir éditer votre code source. Nous proposons de télécharger ECLIPSE IDE. [link : www.eclipse.org](http://www.eclipse.org)



I.1.5 Téléchargement du plugin d'intégration Eclipse-JCDE version 0.2 :

Le plugin d'intégration Eclipse-JCDE version 0.2 : archive **eclipse-jcde-0.2.zip** à partir du site de téléchargement sourceforge :

<http://sourceforge.net/projects/eclipse-jcde/>

 apache-tomcat-10.1.13	10/4/2023 9:06 PM	Application	13,215 KB
 jdk-8u202-windows-x64	10/4/2023 8:56 PM	Application	216,653 KB
 a4059fe1-ab6d-43b4-a3fc-d0ff9ea049c0	10/3/2023 1:28 PM	Chrome HTML Do...	882 KB
 eclipse-jcde-0.2	10/3/2023 9:09 AM	Compressed (zipp...	144 KB
 jre-8u341-windows-x64	10/3/2023 9:08 AM	Application	85,468 KB
 java_card_kit-2_2_2-windows	10/3/2023 9:04 AM	Compressed (zipp...	29,463 KB

I.2 Instructions d'installation :

I.2.1 Installation d'Eclipse sous windows :








Installation du JDK

Remarque:

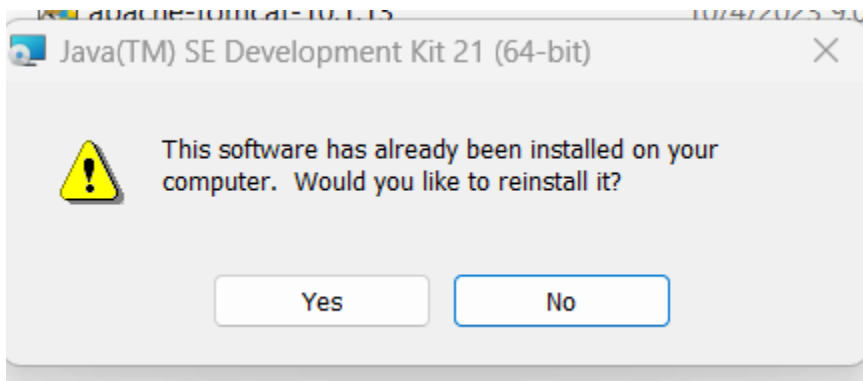
Il ne faut pas confondre le JDK (Java Development Kit) avec le JRE (Java Runtime Environment) :

✓ la JRE ne contient que les outils nécessaires pour exécuter des applications Java;

✓ le JDK permet également d'exécuter des applications Java mais il permet en plus de compiler du code source java pour en faire des applications compréhensible par la JRE, le JRE est inclut dans le JDK.

 eclipse-jcde-0.2	10/3/2023 9:09 AM	Compressed (zipp...	144 KB
 jre-8u341-windows-x64	10/3/2023 9:08 AM	Application	85,468 KB
 java_card_kit-2_2_2-windows	10/3/2023 9:04 AM	Compressed (zipp...	29,463 KB
 apache-tomcat-11.0.0-M11	10/1/2023 6:24 PM	Application	12,191 KB
 eclipse-inst-jre-win64	10/1/2023 6:08 PM	Application	127,188 KB
 android-studio-2022.3.1.20-windows (1)	10/1/2023 9:11 AM	Application	1,096,697 ...
 jdk-21_windows-x64_bin	10/1/2023 8:52 AM	Application	167,721 KB

NB: software already installed



Traitement des variables Environnement

1)

System > About

LAPTOP-E79ED1VU
Vivobook_ASUSLaptop M6500IH_M6500IH

Rename this PC

i

Device specifications

Copy ^

Device name

LAPTOP-E79ED1VU

Processor

AMD Ryzen 7 4800HS with Radeon Graphics2.90 GHz

Installed RAM

8.00 GB (7.42 GB usable)

Device ID

25A7BEE1-AC93-4502-88CE-DA509CCE8945

Product ID

00342-43332-15901-AAOEM

System type

64-bit operating system, x64-based processor

Pen and touch

No pen or touch input is available for this display

Related links

Domain or workgroup

System protection

Advanced system settings

Windows specifications

Copy ^

Edition

Windows 11 Home Single Language

Version

22H2

Installed on

7/30/2023

OS build

22621.2428

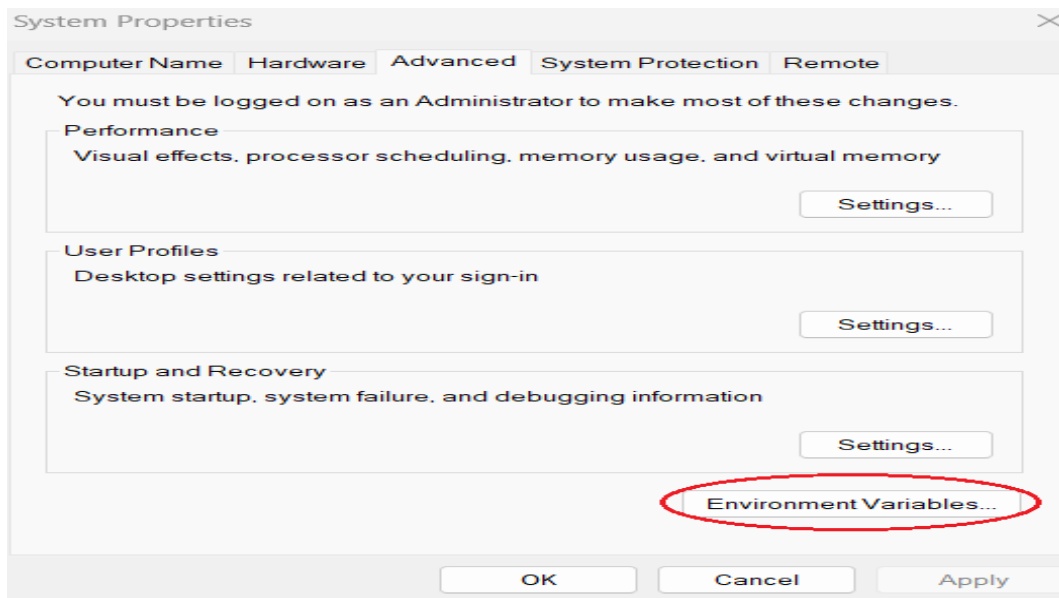
Experience

Windows Feature Experience Pack 1000.22674.1000.0

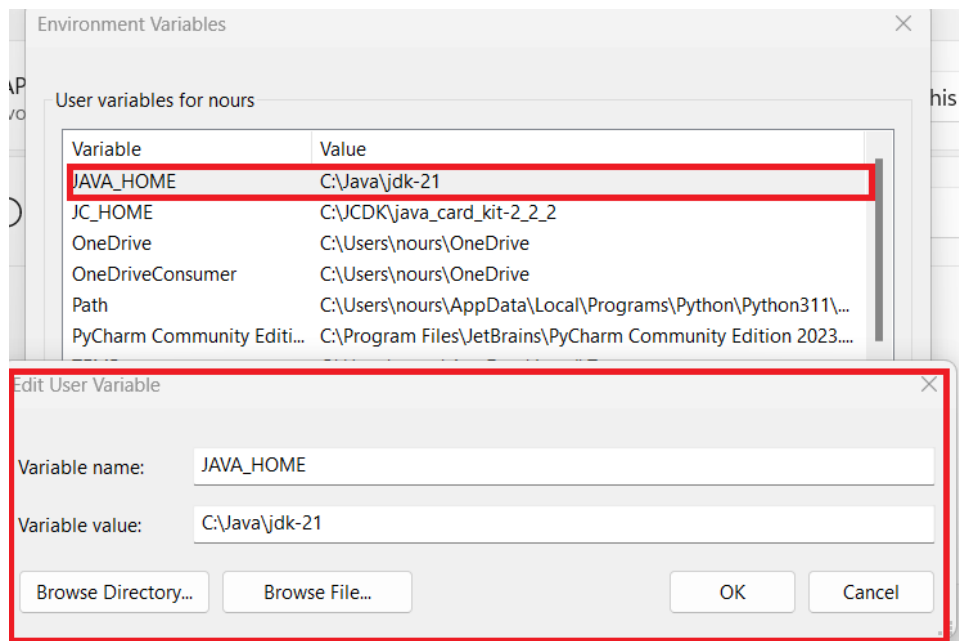
Microsoft Services Agreement

Microsoft Software License Terms

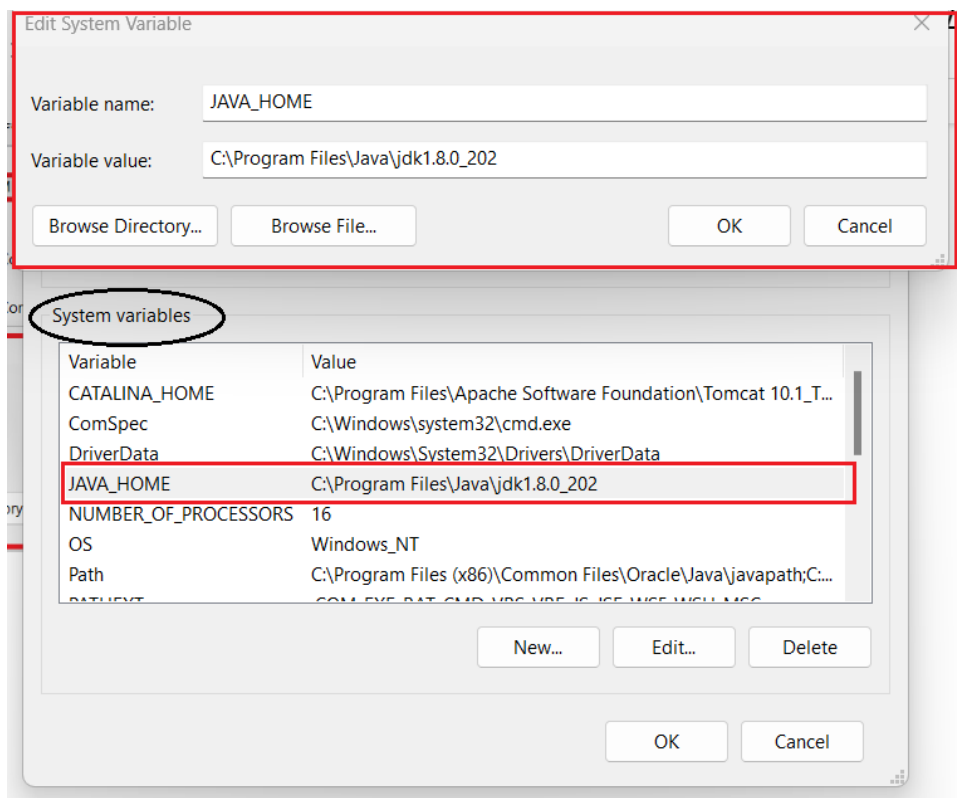
2)



3)



4)



5) Pour vérifier le bon fonctionnement de votre JDK, il suffit de taper la commande « **java – version** » sous la console DOS :

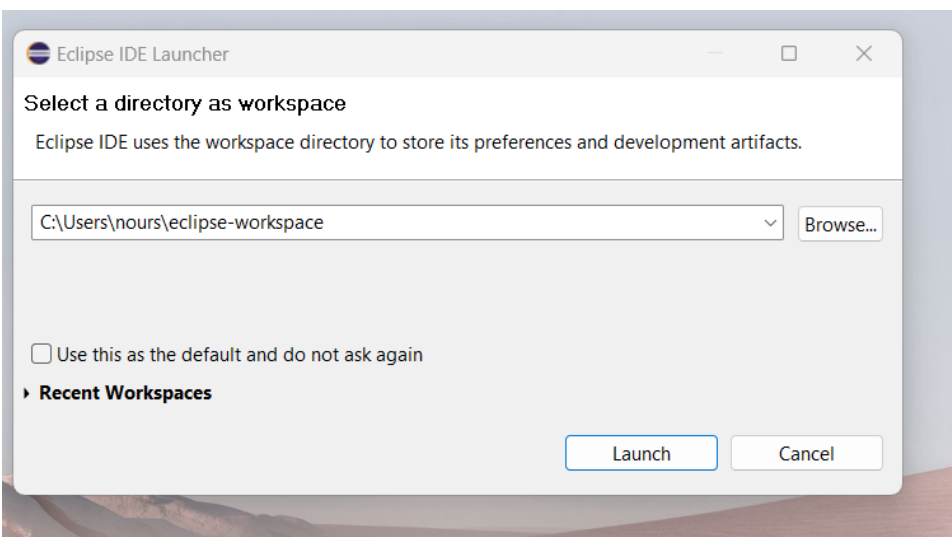
```
C:\Users\nours>java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
C:\Users\nours>
```

Installation de l'IDE - Eclipse :



Remarque :

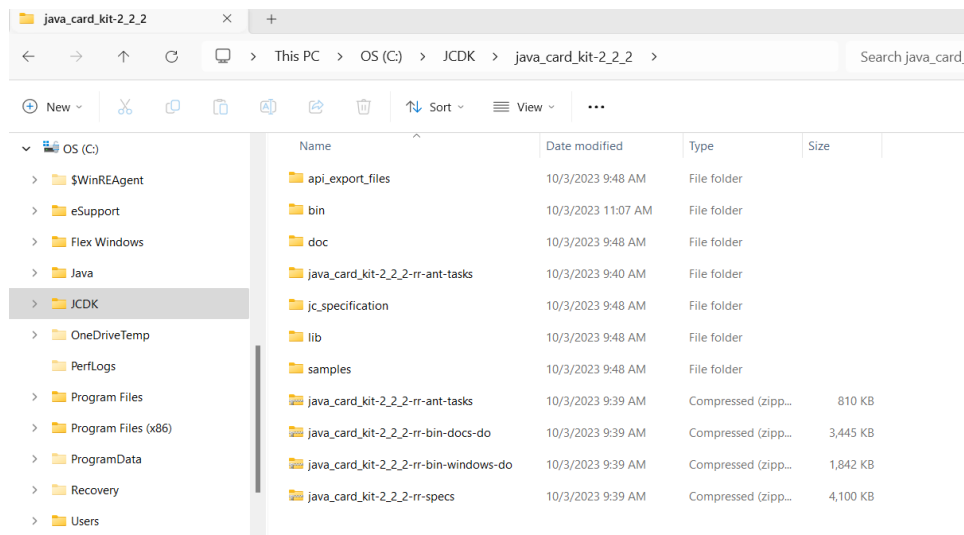
Eclipse vous demande de **choisir un Workspace** (Espace de travail). Notre Workspace sera par défaut le répertoire **C:\Eclipse\workspace** :



I.2.2 Installation du Java Card Development Kit 2 .2.2 :

Tout d'abord, Il faut créer un nouveau répertoire **C:\JCDK** :

- Décompresser l'archive **java_card_kit-2_2_2-windows.zip** après l'avoir téléchargé dans le répertoire **C:\JCDK**.
- Aller dans le répertoire **C:\JCDK \java_card_kit-2_2_2**
- Décompresser toutes les fichiers du répertoire courant **C:\JCDK \java_card_kit-2_2_2** :

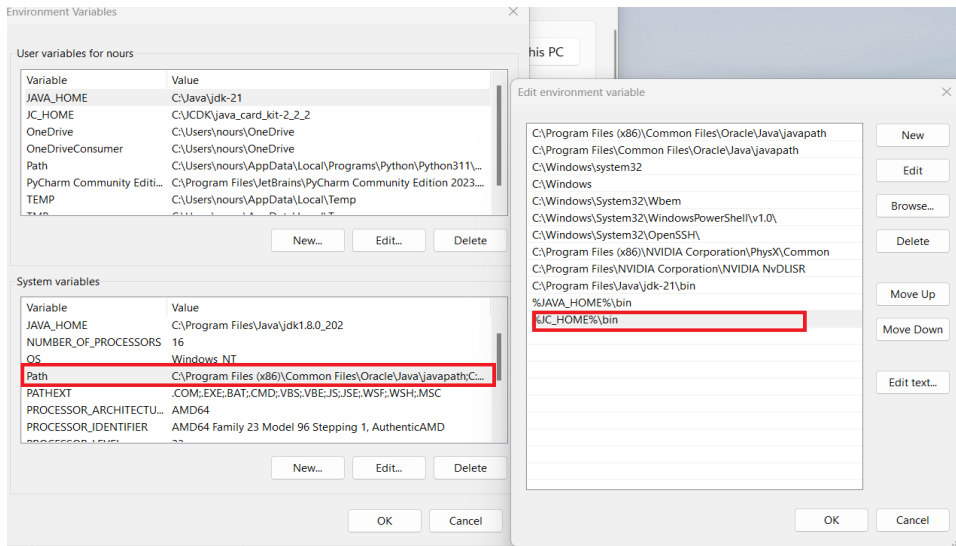


Remarque :

- **C:\JCDK\java_card_kit-2_2_2\bin** : contient les outils du JCDK (simulateur, ...)
- **C:\JCDK\java_card_kit-2_2_2\lib** : contient les librairies du JCDK
- **C:\JCDK\java_card_kit-2_2_2\jc_specifications\specs\api\html** : contient l'API du

JCDK (très utile)

Une fois l'installation terminée, il faut ajouter la variable d'environnement **JC_HOME** contenant le chemin d'accès au JCDK dans la définition de la variable **Path** :



Vérifiez la réussite de configuration en tapant « **apdutool** » dans une console de commandes :

```

Command Prompt
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

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

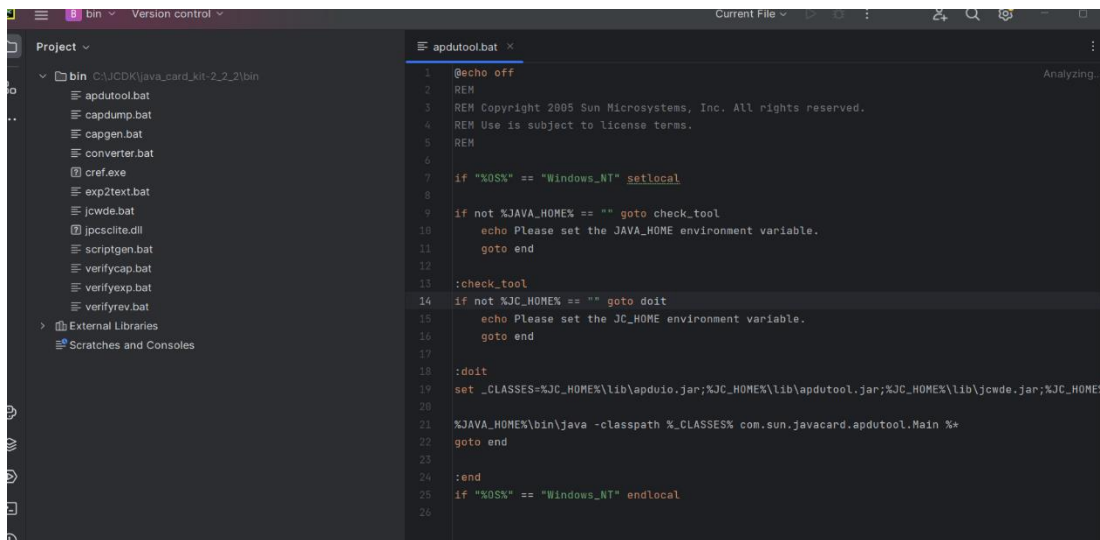
C:\Users\nours>

```

En cas ou la commande apdutool n’a pas fonctionner

On peut réaliser ses étapes suivantes :

1. Accéder au fichier bin situé dans l’arborescence suivante C:\JCDK\java_card_kit-2_2_2\bin
2. Ouvrir le fichier apdutool.bat en mode edit
3. Vérifier que le code est écrit comme ce dessus :



Mise a jour des plugins:

Address bar: This PC > OS (C:) > Users > nous > eclipse > java-2023-09 > eclipse > plugins

Name	Date modified	Type	Size
.DS_Store	10/16/2023 10:13 PM	DS_STORE File	7 KB
org.eclipse.equinox.launcher_1.6.500.v202...	10/1/2023 6:14 PM	Executable Jar File	55 KB
org.eclipse.jdt.core_2.0	10/16/2023 10:13 PM	Executable Jar File	48 KB
org.eclipse.jdt.core.ref_0.1.0	10/16/2023 10:13 PM	Executable Jar File	9 KB
org.eclipse.jdt.core.jcbp.wizards_0.1.0	10/16/2023 10:13 PM	Executable Jar File	9 KB
org.eclipse.jdt.core.jcbp_0.1.0	10/16/2023 10:13 PM	Executable Jar File	22 KB
org.eclipse.jdt.core.jctools_0.1.0	10/16/2023 10:13 PM	Executable Jar File	21 KB
org.eclipse.jdt.core.jcwe_0.1.0	10/16/2023 10:13 PM	Executable Jar File	10 KB
org.eclipse.jdt.preferences_0.1.0	10/16/2023 10:13 PM	Executable Jar File	8 KB
org.eclipse.jdt.wizards_0.1.0	10/16/2023 10:13 PM	Executable Jar File	34 KB

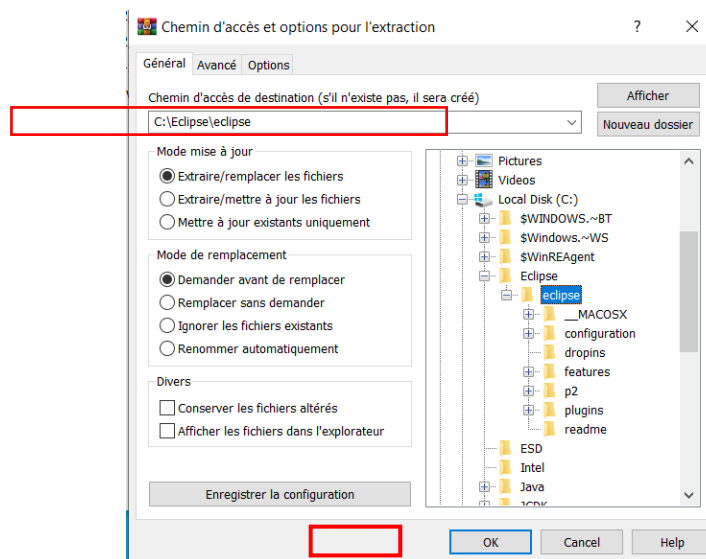
C'est un ensemble des méthodologies, des outils et des concepts informels imposés par les normes adaptées à l'environnement d'utilisation afin de favoriser la production et la maintenance de composants logiciels de qualité.

Le génie logiciel est généralement réservé aux logiciels complexes de grande envergure et non aux applications ou programmes simples. Le développement n'est toutefois qu'une phase du processus. Les ingénieurs logiciels sont responsables de la conception des systèmes, alors que les programmeurs sont chargés du codage permettant leur implémentation.

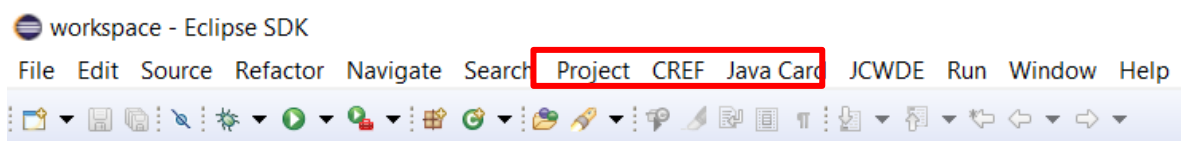
1- Télécharger le zip **eclipse-jcde-0.2.zip** à partir du lien suivant :

<https://sourceforge.net/projects/eclipse-jcde/files/eclipse-jcde/eclipse-jcde-0.2.zip/download>

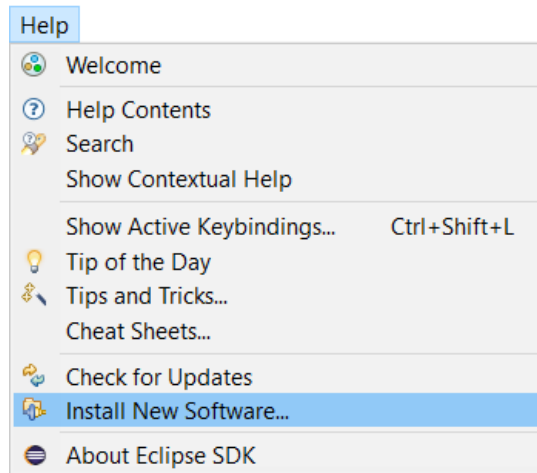
2- Décompresser l'archive **eclipse-jcde-0.2.zip** après l'avoir téléchargé dans le répertoire **C:\Eclipse\eclipse**



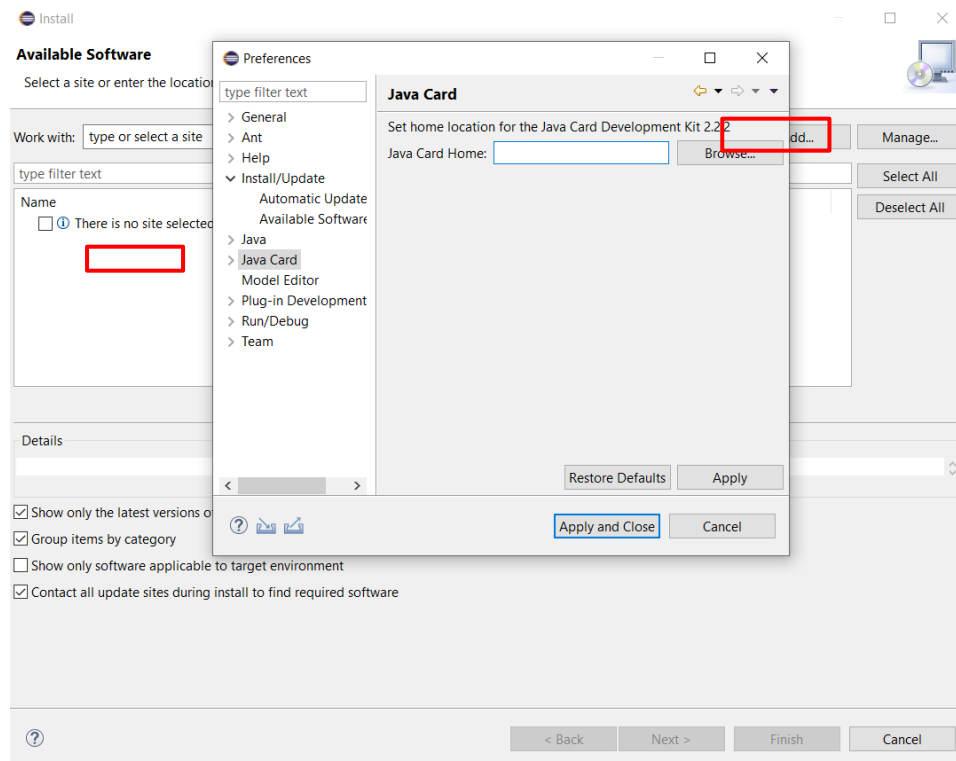
3- Ouvrir Eclipse, vous remarquerez que les plugins sont ajoutés dans la barre de menu.



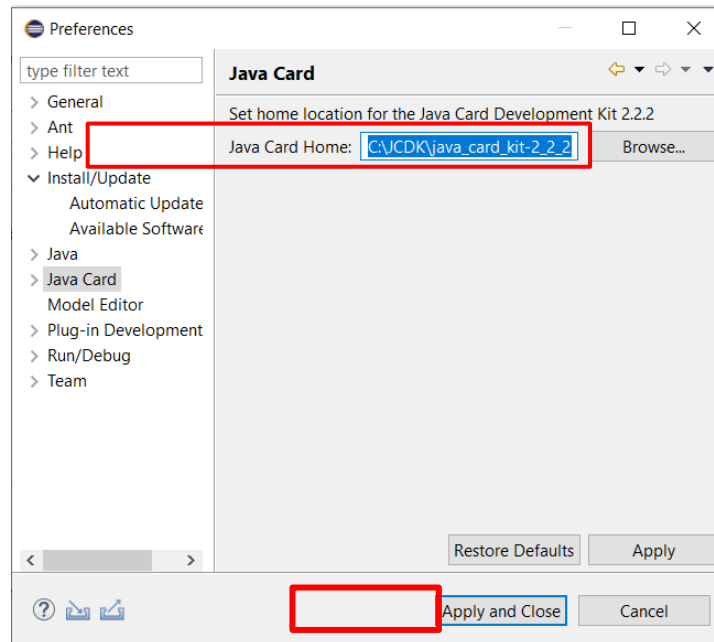
4- Maintenant allez sur **Help>Install New Software...**



5- Cliquez sur **Manage** puis **JavaCard**.



6- Ajouter le chemin de Java Card Home **C:\JCDK\java_card_kit-2_2_2** puis cliquer sur **Apply and Close**.



I. Conclusion :

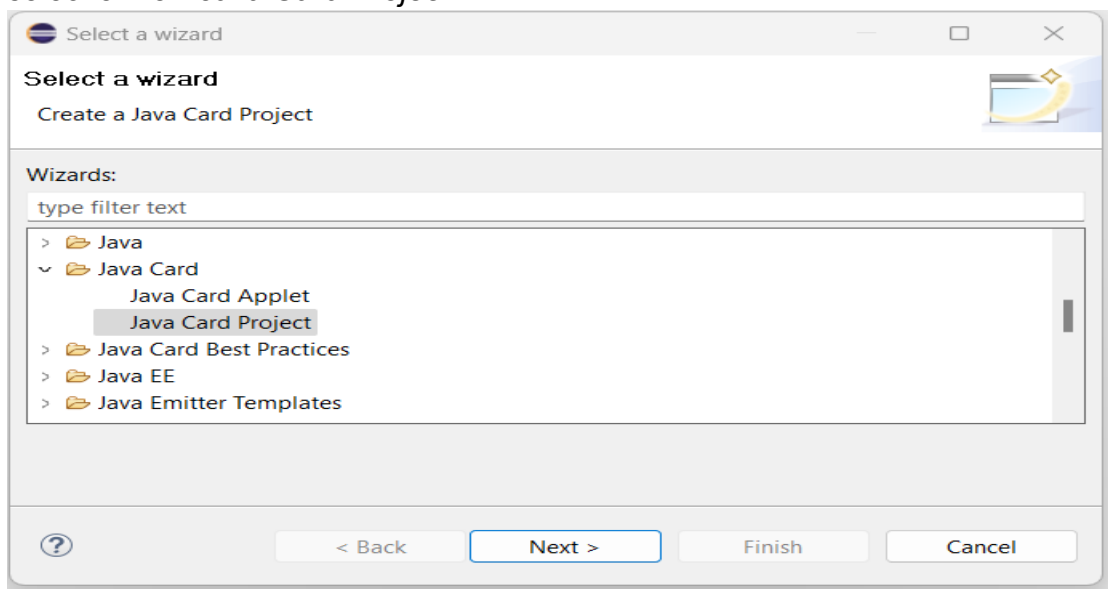
A ce stade, nous disposons d'un environnement de développement complet permettant de créer des Applet Javacard, de les simuler et de créer des applications clientes.

Chapitre II. Développement d'une application coté Serveur

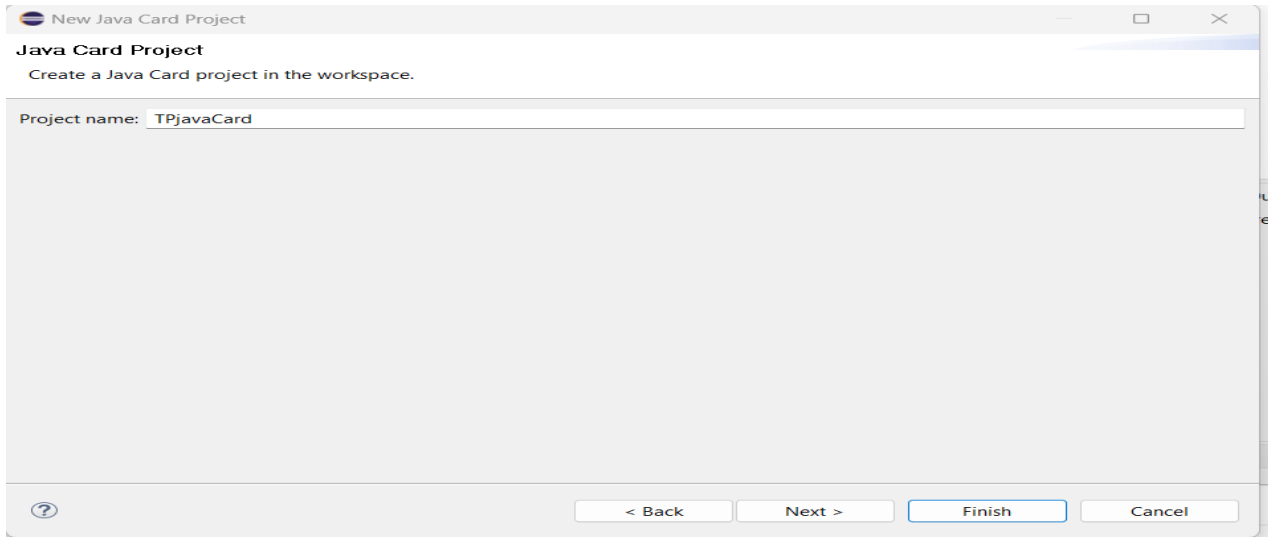
II. 1)Creation de l'applet card sous Eclipse :

1.1. Création d'un nouveau projet :

- Lancer Eclipse, dans le menu File, faire New puis Other et sélectionner "*Java Card Project* ":

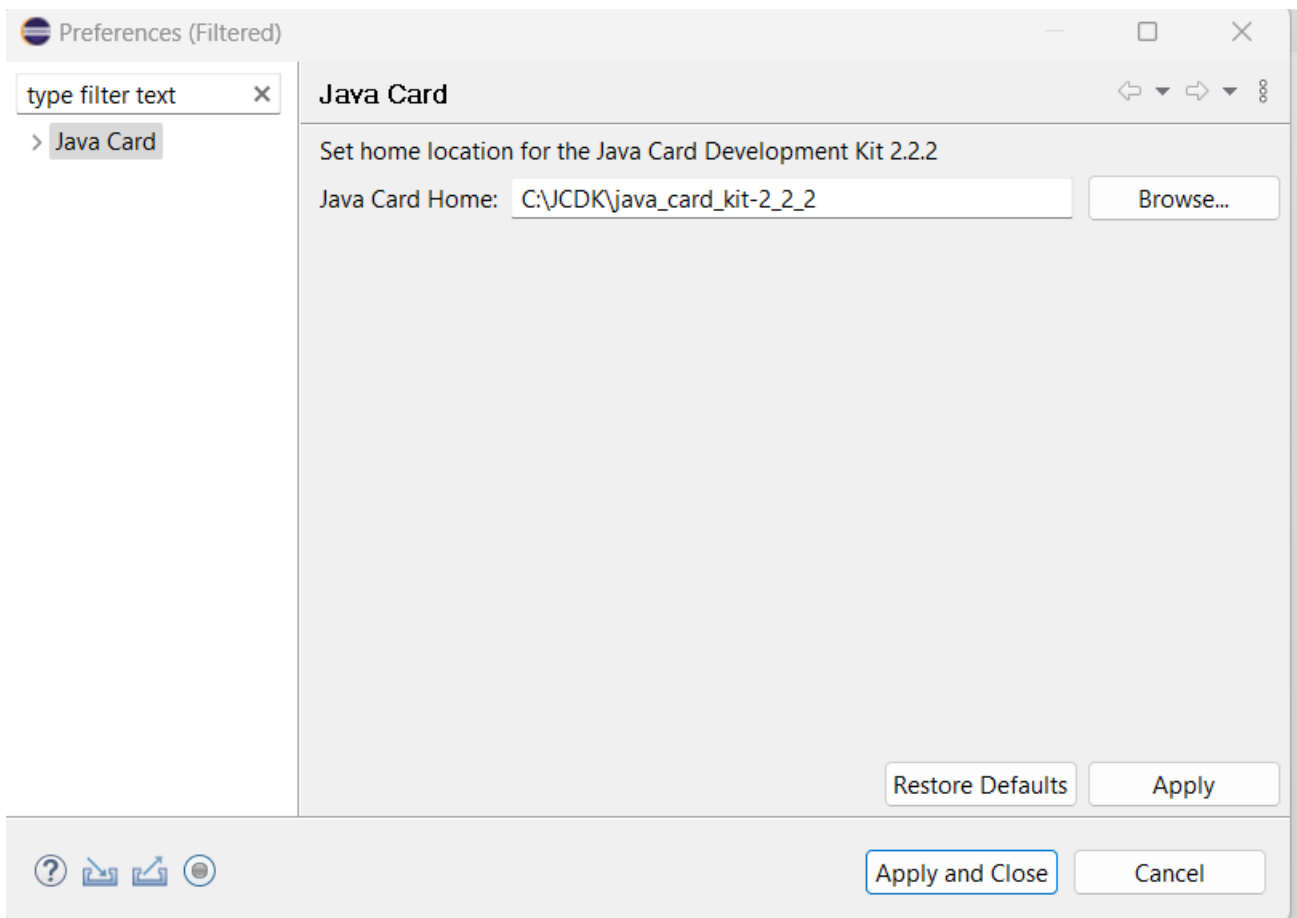


- Cliquer alors sur le bouton **Next**. Donner alors un nom au projet. Nous l'appellerons "**TP Javacard**"



- Cliquer alors sur le bouton **Finish**. Vous devriez alors voir le message d'erreur

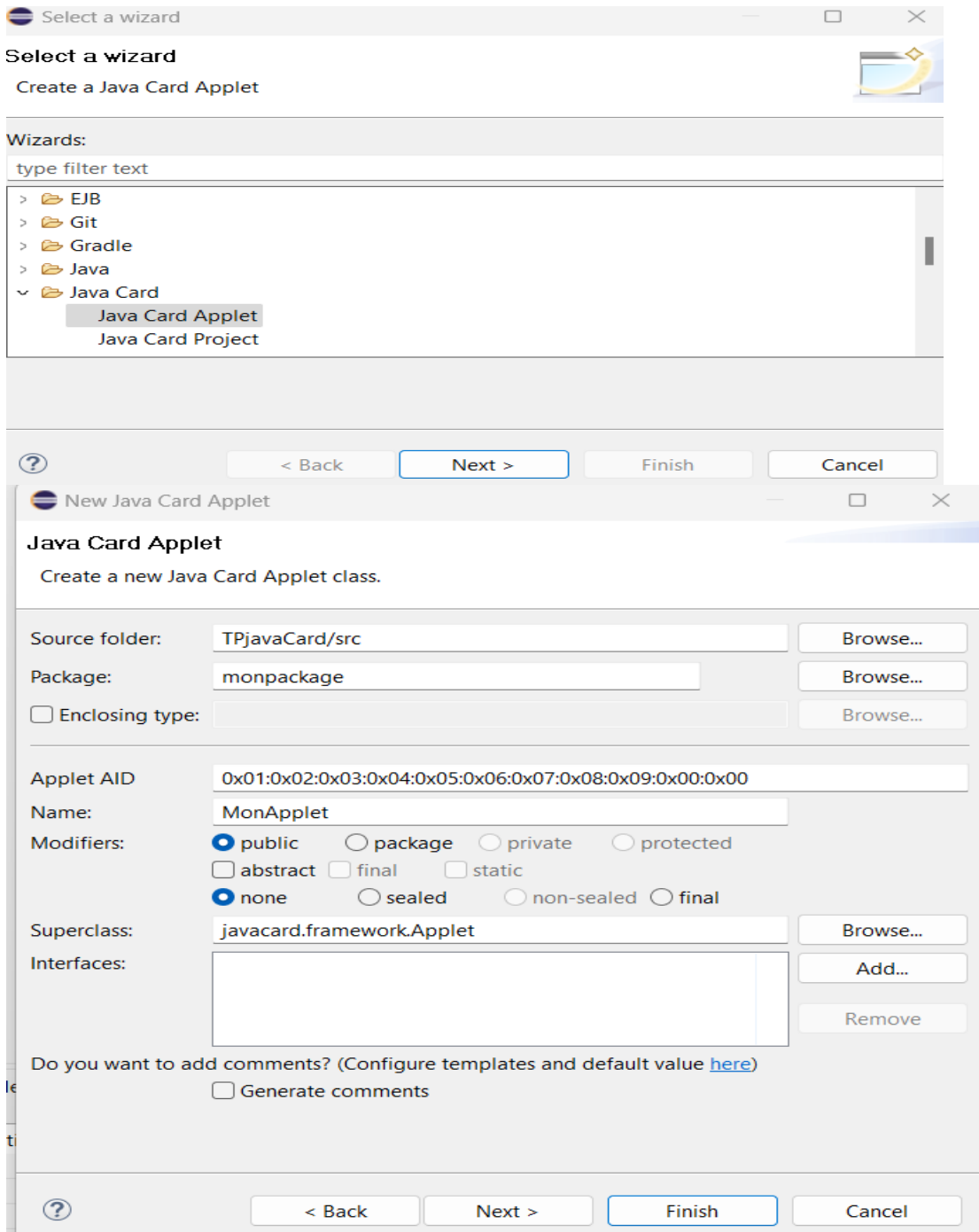
Entrer alors le dossier où a été installé le Java Card Development Kit 2.2.2, puis cliquer sur le bouton **OK** :



1.2. Création d'une applet Javacard:

- Aller dans le menu **File**, faire **New** puis **Other...** Sélectionner alors "*Java Card Applet*" puis cliquer sur le bouton **Next**.

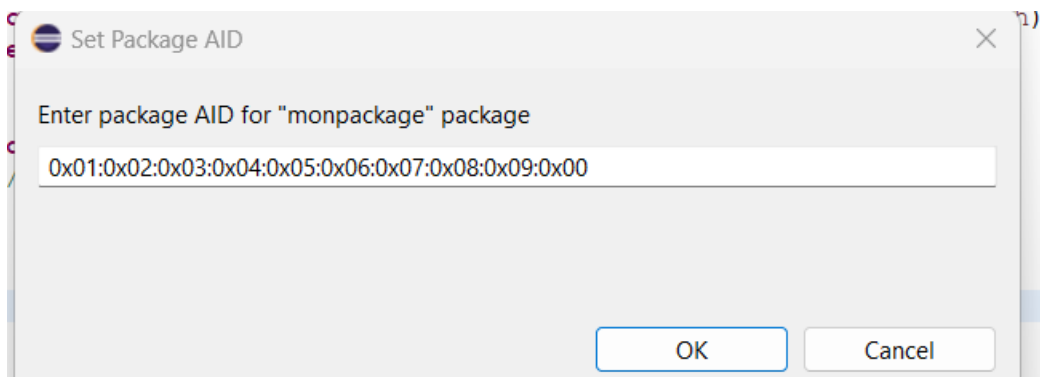
Donner alors un nom au package ainsi qu'à l'applet. Nous les appellerons respectivement "monpackage" et "MonApplet" :



Cliquer alors sur le bouton **Finish**. Eclipse vient de générer automatiquement le squelette de notre Applet :

```
*MonApplet.java ×
1 package monpackage;
2
3 import javacard.framework.APDU;
4
5
6
7 public class MonApplet extends Applet {
8
9     private MonApplet() {
10    }
11
12     public static void install(byte bArray[], short bOffset, byte bLength) throws ISOException {
13         new MonApplet().register();
14     }
15
16     public void process(APDU arg0) throws ISOException {
17         // TODO Auto-generated method stub
18     }
19 }
```

- Clic droit sur le package monpackage, **Java Card Tools, Set Package AID** :



II.2 Codage de notre applet :

Nous allons créer une applet **compteur**. L'applet comportera 4 fonctions :

- incrémenter le compteur
- décrémenter le compteur
- interroger le compteur
- initialiser le compteur à une valeur donnée

Étape 1. Ajouter API JavaCard :

```

1 package monpackage;
2
3 import javacard.framework.APDU;
4 import javacard.framework.Applet;
5 import javacard.framework.ISO7816;
6 import javacard.framework.ISOException;
7

```

Étape 2. Déclarer les attributs et les constantes :

```

8 public class MonApplet extends Applet {
9
10     /* Constantes */
11     public static final byte CLA_MONAPPLET = (byte) 0xB0;
12     public static final byte INS_INCREMENTER_COMPTEUR = 0x00;
13     public static final byte INS_DECREMENTER_COMPTEUR = 0x01;
14     public static final byte INS_INTERROGER_COMPTEUR = 0x02;
15     public static final byte INS_INITIALISER_COMPTEUR = 0x03;
16     /* Attributs */
17     private byte compteur;
18

```

Étape 3. Définition des méthodes publiques qu'elle doit obligatoirement implémenter

On définit tout d'abord :

1) la méthode install () : création et enregistrement de l'objet Applet

```

/* Constructeur */
private MonApplet() {
    compteur = 0;
}

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

```

2) la méthode process () : Traitement des commandes APDU

```

MonApplet.java x
21     compteur = 0;
22 }
23 public static void install(byte bArray[], short bOffset, byte bLength)
24     throws ISOException {
25     new MonApplet().register();
26 }
27 public void process(APDU apdu) throws ISOException {
28     byte[] buffer = apdu.getBuffer();
29     if (this.selectingApplet()) return;
30     if (buffer[ISO7816.OFFSET_CLA] != CLA_MONAPPLET) {
31         ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);
32     }
33     switch (buffer[ISO7816.OFFSET_INS]) {
34     case INS_INCREMENTER_COMPTEUR:
35         compteur++;
36         break;
37     case INS_DECREMENTER_COMPTEUR:
38         compteur--;
39         break;
40     case INS_INTERROGER_COMPTEUR:
41         buffer[0] = compteur;
42         apdu.setOutgoingAndSend((short) 0, (short) 1);
43         break;
44     case INS_INITIALISER_COMPTEUR:
45         apdu.setIncomingAndReceive();
46         compteur = buffer[ISO7816.OFFSET_CDATA];
47         break;
48     default: ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
49     }
50 }
51 }
52 }
53 }
54 }}

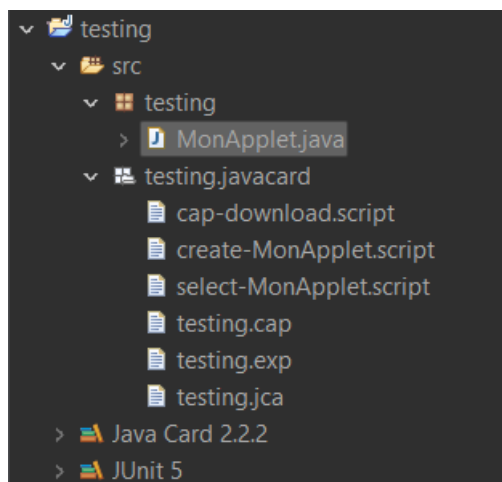
```

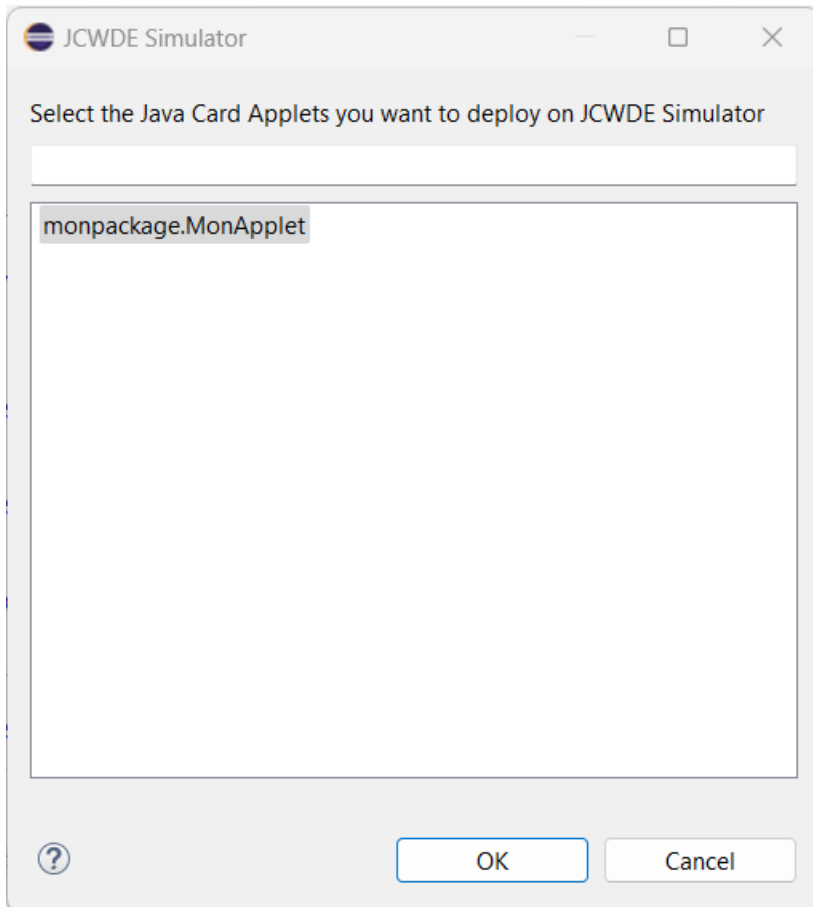
III.3 Outils de simulation :

3.1.JCWDE : simulateur sans conservation d'état

On fait un clic droit sur le package, sélectionner Java Card Tools puis Generate Script. Ceci a pour conséquence de générer automatiquement les APDU nécessaires à l'upload, l'instanciation (installation) et la sélection de l'applet sur une Javacard. testing.javacard contient alors 3 scripts :

- cap-downloadsript
- create-MonAppletscript
- select-MonApplet.script





lancer l'outil **APDUTOOL**:

```
C:\Users\nours>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.
```

- Installons notre applet en recopiant les APDU contenus dans le script **createMonApplet.script**
- sélectionner notre applet en recopiant l'APDU contenu dans **selectMonApplet.script**

```
Command Prompt - apdutool x + v
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nours>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 0x00
// Select the installer applet
0x00 0xA4 0x04 0x00 0x09 0xA0 0x00 0x00 0x00 0x62 0x03 0x01 0x08 0x01 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
// create MonApplet applet
0x80 0xB8 0x00 0x00 0xd 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x00 0x7F;
CLA: 80, INS: b8, P1: 00, P2: 00, Lc: 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, 00, Le: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00, SW1: 90, SW2: 00
powerup;
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
// select MonApplet applet
0x00 0xA4 0x04 0x00 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 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, 00, SW1: 90, SW2: 00
```

- tester l'applet. Commençons par **interroger le compteur (INS = 0x02)** en envoyant l'APDU:

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

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

- **incrémenter le compteur (INS = 0x00)** en envoyant l'APDU:

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

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

- **Interrogeons de nouveau le compteur (INS = 0x02)** en envoyant l'APDU:

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

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

- **Initialisons** maintenant (INS = 0x03) le compteur à 0x4A en envoyant l'APDU suivante

0xB0 0x03 0x00 0x00 0x01 0x4A 0x7F;

```
0xB0 0x03 0x00 0x00 0x01 0x4A 0x7F;
```

```
CLA: b0, INS: 03, P1: 00, P2: 00, Lc: 01, 4a, Le: 00, SW1: 90, SW2: 00
```

- **Décrémentons le compteur** (INS = 0x01) en envoyant l'APDU suivante :

0xB0 0x01 0x00 0x00 0x00 0x7F;

```
0xB0 0x01 0x00 0x00 0x00 0x7F;
```

```
CLA: b0, INS: 01, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
```

- **Interrogeons de nouveau** le compteur (INS = 0x02) en envoyant l'APDU suivante :

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

```
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 49, SW1: 90, SW2: 00
```

- Nous allons nous déconnecter du simulateur en tapant "**powerdown;**" dans **apdutool**, ce qui provoque **la fermeture de JCWDE** dans la console d'Eclipse :
powerdown;

```

Command Prompt - apdutool x + v - □ ×
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 0
9, 00, 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, 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
0xB0 0x02 0x00 0x00 0x00 0x7F;
CLA: b0, INS: 02, P1: 00, P2: 00, Lc: 00, Le: 01, 01, SW1: 90, SW2: 00
0xB0 0x03 0x00 0x00 0x01 0x4A 0x7F;
CLA: b0, INS: 03, P1: 00, P2: 00, Lc: 01, 4a, Le: 00, 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, 49, SW1: 90, SW2: 00
powerdown;

```

3.2 CREF : simulateur avec conservation d'état:

- Nous pouvons lancer **CREF** et créer notre fichier image. Pour cela, ouvrir une invite de commandes, lancer le script **setvars.bat**, puis taper "**cref -o monapplet.eeprom**" et valider :

```

Command Prompt - cref -o n x + v - □ ×
C:\Users\nours>cref -o 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    0xfffff
  E2P       0x10020    0xffe0    0x1fffff

  ROM Mask size =          0xce64 =          52836 bytes
  Highest ROM address in mask = 0xee63 =          61027 bytes
  Space available in ROM =    0x119c =          4508 bytes
EEPROM will be saved in file "monapplet.eeprom"
Mask has now been initialized for use

```

- Uploadons maintenant notre applet. Pour cela, clic droit sur le package monpackage, **Java Card Tools, Deploy** :

```

terminated> C:\Program Files\Java\jdk1.5.0_22\bin\javaw.exe (Oct 31, 2023, 3:56:51 AM) [pid: 3640]
CLA: 80, INS: b2, P1: 07, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 07, P2: 00, Lc: 20, 07, 00, 7d, 00, 02, 10, 18, 8c, 00, 01, 18, 03, 88, 00, 7a, 02, 30, 8f, 00, 02, 3d, 8c, 00,
CLA: 80, INS: b4, P1: 07, P2: 00, Lc: 20, 00, 05, 2d, 18, 8b, 00, 06, 60, 03, 7a, 1a, 03, 25, 10, b0, 6a, 08, 11, 6e, 00, 8d, 00, 07,
CLA: 80, INS: b4, P1: 07, P2: 00, Lc: 20, 03, 00, 0f, 00, 1a, 00, 25, 00, 32, 18, 3d, 84, 00, 04, 41, 5b, 88, 00, 70, 2d, 18, 3d, 84,
CLA: 80, INS: bc, P1: 07, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b2, P1: 08, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 08, P2: 00, Lc: 0d, 08, 00, 0a, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: bc, P1: 08, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b2, P1: 05, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 05, P2: 00, Lc: 20, 05, 00, 2a, 00, 0a, 02, 00, 00, 00, 06, 80, 03, 00, 01, 00, 00, 00, 06, 00, 00, 01, 03, 80,
CLA: 80, INS: b4, P1: 05, P2: 00, Lc: 0d, 03, 06, 80, 07, 01, 03, 80, 0a, 08, 03, 80, 0a, 06, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: bc, P1: 05, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b2, P1: 09, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: b4, P1: 09, P2: 00, Lc: 18, 09, 00, 15, 00, 07, 0a, 3f, 05, 06, 05, 06, 14, 00, 0a, 05, 0a, 04, 03, 07, 05, 10, 33, 06,
CLA: 80, INS: bc, P1: 09, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00
CLA: 80, INS: ba, P1: 00, P2: 00, Lc: 00, Le: 00, SW1: 90, SW2: 00

```

- Relançons CREF afin d'installer notre applet, en prenant soin de recharger notre fichier image (option -i) :

```

Command Prompt - cref -i m
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

```

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

```
Problems Terminal Data Source Explorer Properties Console ×
<terminated> C:\Program Files\Java\jdk1.5.0_22\bin\javaw.exe (Oct 31, 2023, 4:02:35 AM) [pid: 11016]
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license
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: 0
CLA: 80, INS: b8, P1: 00, P2: 00, Lc: 0d, 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0
```

- Puis, dans un autre terminal, **lançons apdutool**, sélectionnons notre applet, après quoi nous pouvons envoyer des APDU à notre applet :

```
Command Prompt - apdutool × + ▾
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

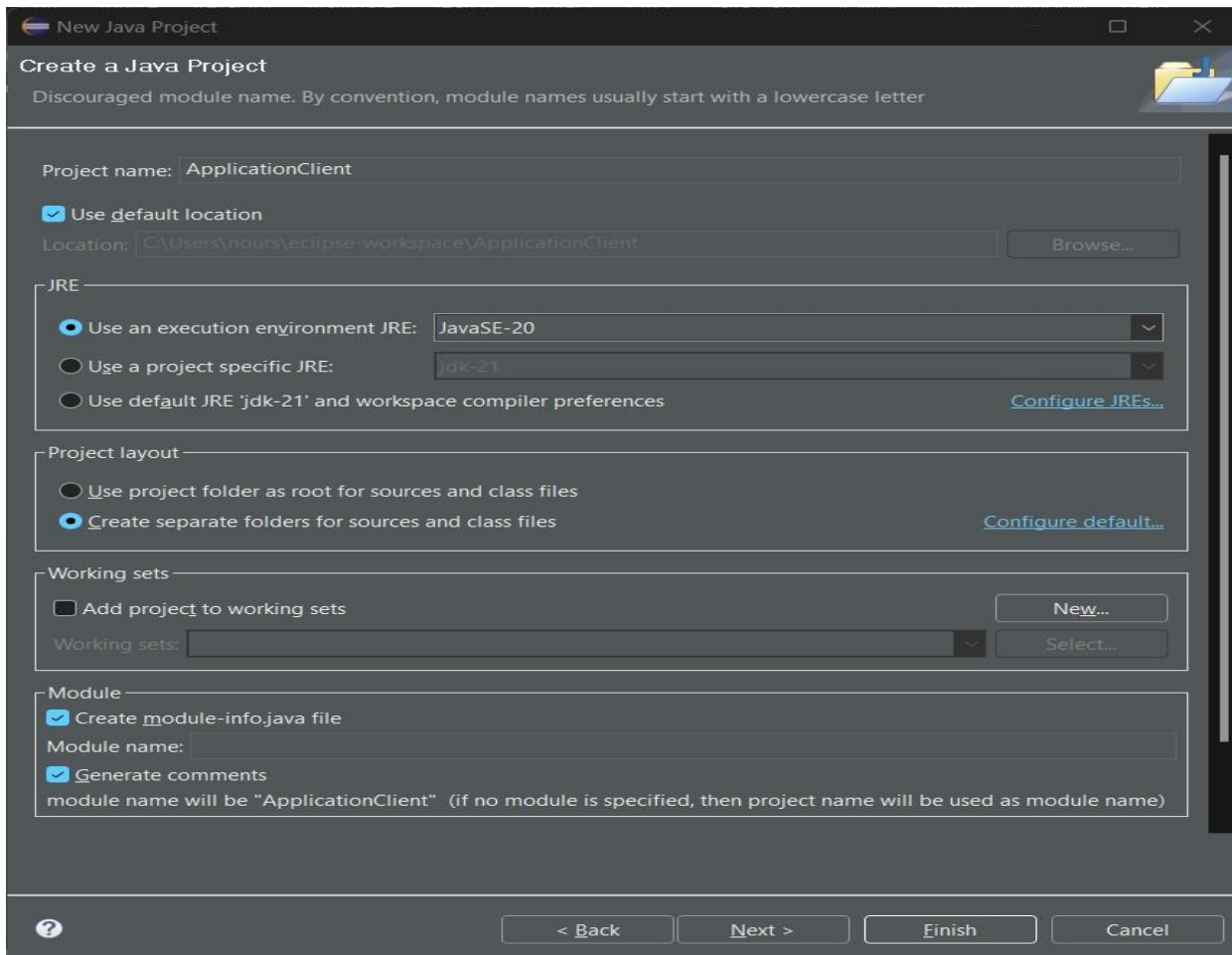
C:\Users\nours>apdutool
Java Card 2.2.2 APDU Tool, Version 1.3
Copyright 2005 Sun Microsystems, Inc. All rights reserved. Use is subject to license
terms.
Opening connection to localhost on port 9025.
Connected.
powerup;
Received ATR = 0x3b 0xf0 0x11 0x00 0xff 0x00
// select MonApplet applet
0x00 0xA4 0x04 0x00 0xb 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x00 0x00 0x7F;
CLA: 00, INS: a4, P1: 04, P2: 00, Lc: 0b, 01, 02, 03, 04, 05, 06, 07, 08, 09, 00, 00,
Le: 00, SW1: 6d, SW2: 00
```

Chapitre III. Programmation d'une application coté client

III.1 Création de l'application client sous Eclipse :

III.1.1 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** :



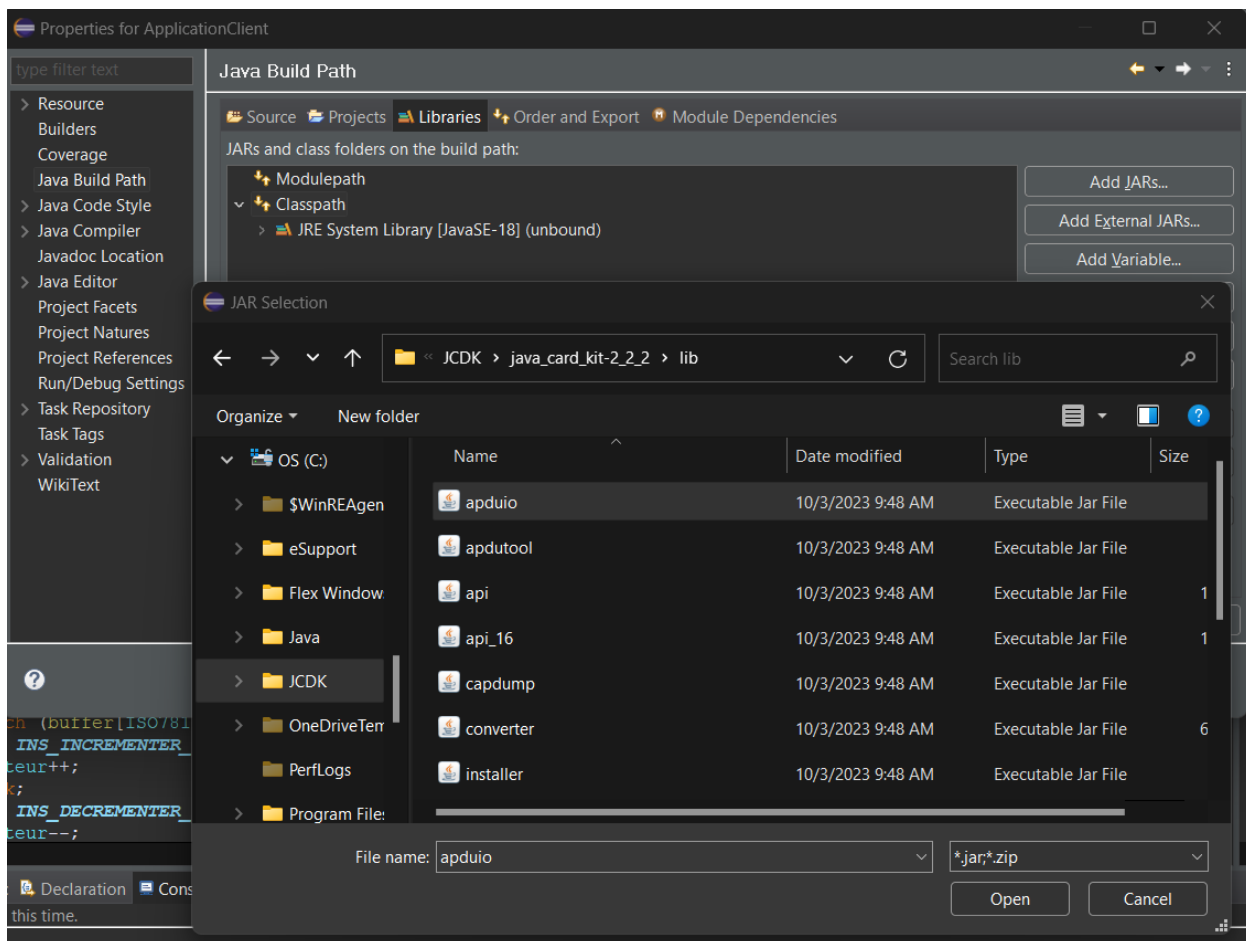
III.1.2 Ajout de la librairie « apduio » dans le classpath

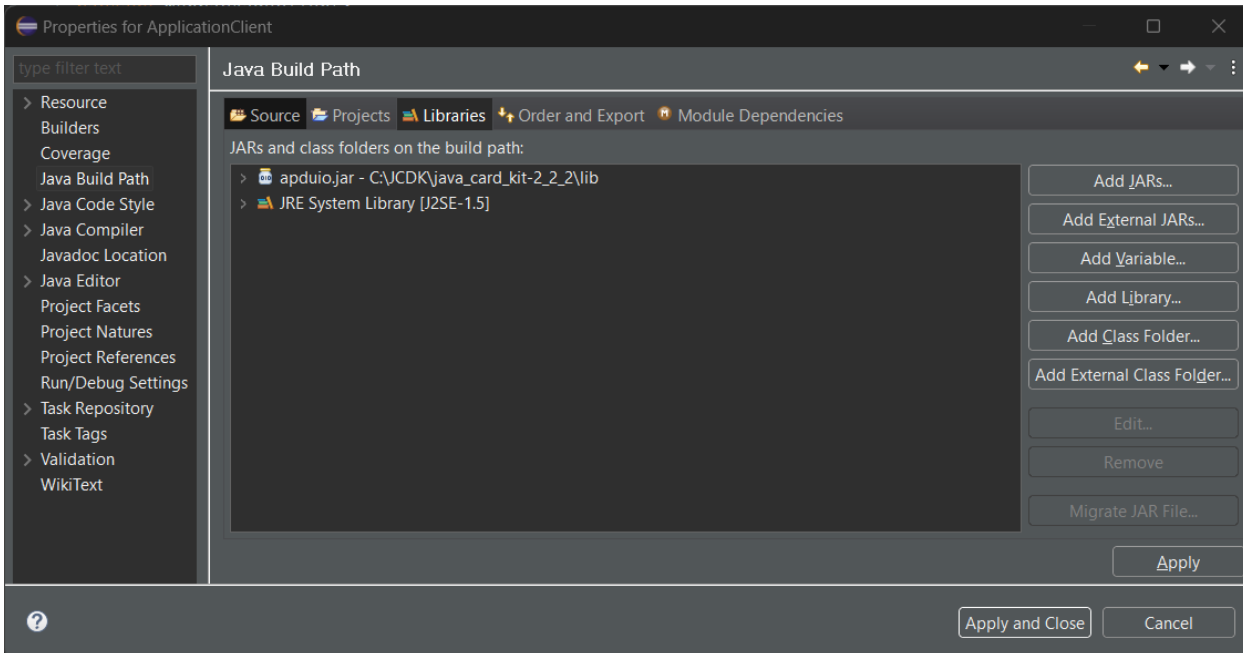
Afin de pouvoir utiliser les classes servant à communiquer avec notre Javacard, il faut ajouter la bibliothèque **apduio.jar** (présente dans le répertoire **C:\JCDK\java_card_kit-2_2_2\lib**).

- Pour cela, faire un clic droit sur notre projet **Application Client** puis **Propriétés...**

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

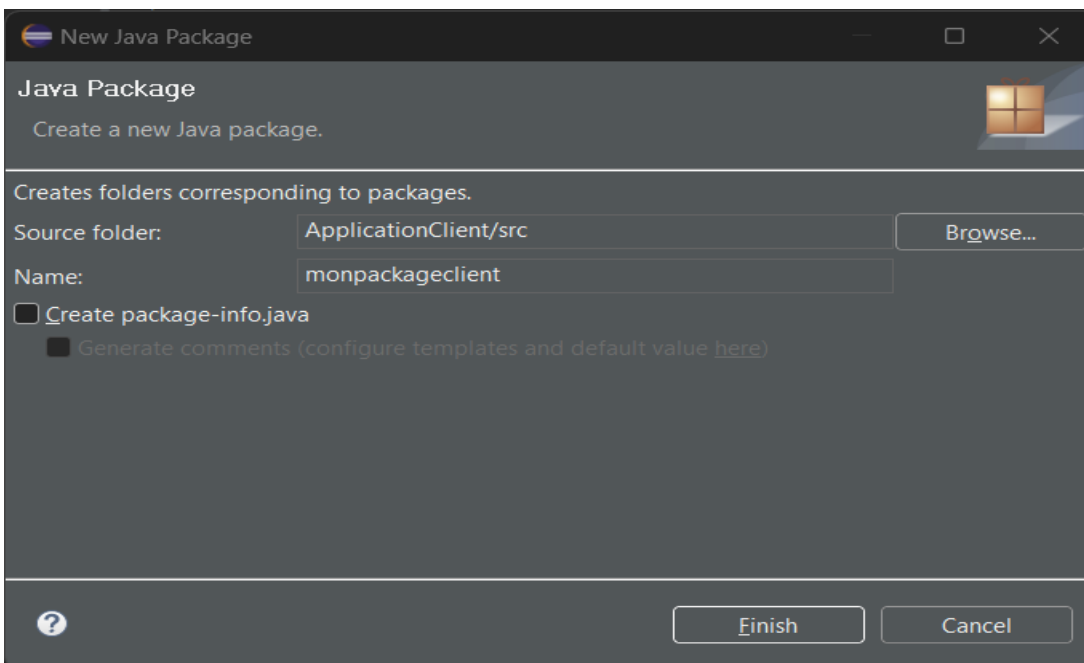
- Sélectionner alors le **fichier apduio.jar**, valider et appuyer sur le bouton **OK**





III.1.3 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** :

New Java Class

Java Class

Create a new Java class.

Source folder: ApplicationClient/src Browse...

Package: monpackageclient Browse...

☐ Enclosing type: Browse...

Name: Maclasse

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

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☒ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

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

Finish Cancel

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 :

La connexion au simulateur se fait via une socket. Le simulateur écoute par défaut sur le **port 9025**. La classe que nous utiliserons pour les échanges de données est **CadT1Client**.

```
// TODO Auto-generated method stub
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;
}
/* 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

La **sélection d'applet** se fait en envoyant la commande SELECT APDU (voir annexe A)

```
/* Sélection de l'applet */
Apdu apdu = new Apdu();
apdu.command[Apdu.CLA] = 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);
}
```

Etape 3 - Invocation des services implémentés

Il suffit pour chaque opération **d'initialiser** correctement **une instance de l'objet APDU** et de l'envoyer à **la carte** via l'instance de la classe **CadT1Client**

```
/* 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 - Incrémenter le compteur");
    System.out.println("3 - Décrémenter le compteur");
    System.out.println("4 - Réinitialiser le compteur");
    System.out.println("5 - Quitter");
    System.out.println();
    System.out.println("Votre choix ?");
    int choix = System.in.read();
    while (!(choix >= '1' && choix <= '5')) {

        choix = System.in.read();
    }
    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':
            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]);}
            break;
        case '2':
            apdu.command[Apdu.INS] =
```

```

case '2':
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");}
break;
case '3':
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");
}
break;

case '4':
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");
}
break;
case '5':
fin = true;
break;

}

```

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;}
}

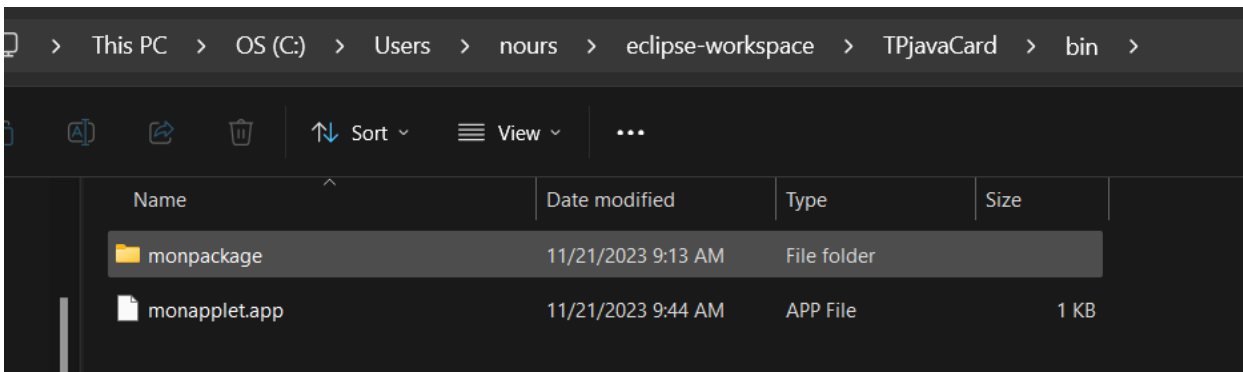
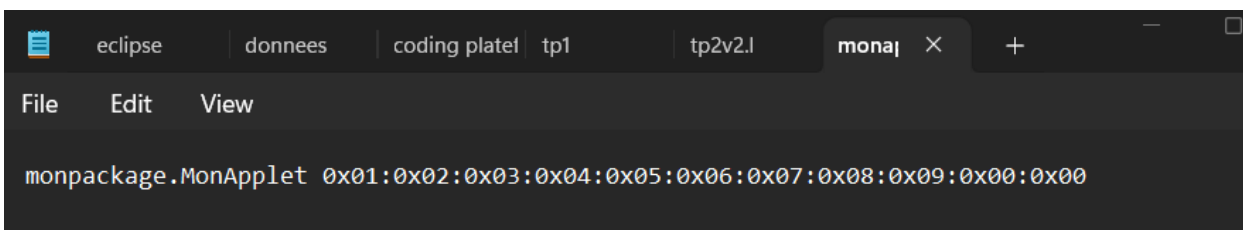
```

III.2 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 card « `.\workspace\TP Javacard\bin` ») . Il contiendra la ligne suivante :

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



```
C:\Users\nours\eclipse-workspace\TPjavaCard\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** :

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

```
Problems Javadoc Declaration Console × Servers
Maclasse [Java Application] C:\Program Files\Java\jdk1.5.0_22\bin\javaw.exe

Application cliente Javacard
-----

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

Votre choix ?
1
Valeur du compteur : 0

Application cliente Javacard
-----
```

```
Votre choix ?
1
Valeur du compteur : 3

Application cliente Javacard
-----

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

Votre choix ?
```

```
Problems Javadoc Declaration Console × Servers
Maclasse [Java Application] C:\Program Files\Java\jdk1.5.0_22\bin\javaw.e
OK

Application cliente Javacard
-----

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

Votre choix ?
1
Valeur du compteur : 2

Application cliente Javacard
-----
```

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 lecompteur
5 - Quitter
```

```
Votre choix ?
```

```
5
```

Command Prompt

```
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\nours>cd C:\Users\nours\eclipse-workspace\TPjavaCard\bin
```

```
C:\Users\nours\eclipse-workspace\TPjavaCard\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.
```

```
jcwde exiting on receipt of power down command.
```



# Chapitre IV. Réalisation du Mini-Projet

---

## IV.1 Interface Graphique ( Partie Client ) :

Pour réaliser l'interface graphique du Partie Client, on va utiliser Java Swing.

Il existe deux méthodes pour la réalisation du design de cette interface :

- Code pure
- En utilisant la palette (Drag and Drop)

⇒ Dans ce projet on va construire l'interface en utilisant la palette. Pour cela on va installer le plugin **WindowBuilder** dans eclipse.

### IV.1.1 WindowBuilder Introduction :

Le plugin Eclipse WindowBuilder est un concepteur Java GUI visuel, puissant et facile à utiliser permettant la création d'applications GUI Java sans vous casser la tête à écrire du code pour afficher des objets graphiques simples comme fenêtres, bouton de commandes, champs de textes... Avec WindowBuilder, vous pouvez créer des fenêtres compliquées en quelques minutes, il suffit d'utiliser le concepteur visuel et le code Java sera automatiquement généré pour vous. Vous pouvez facilement ajouter des contrôles à l'aide de glisser-déposer, gérer les événements de vos contrôles, modifier diverses propriétés des contrôles à l'aide d'un éditeur de propriétés et bien plus encore. Le code généré ne nécessite aucune bibliothèque personnalisée supplémentaire pour compiler et exécuter : l'ensemble du code généré peut être utilisé sans installer WindowBuilder.

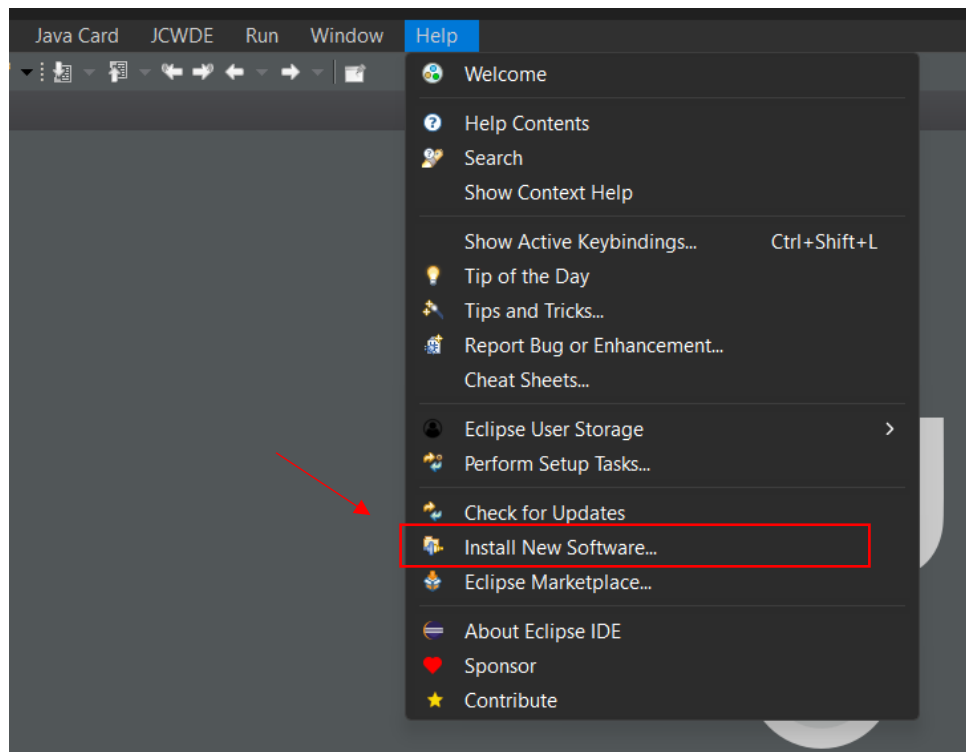
L'éditeur est doté des principaux composants interface utilisateur (user interface ) suivants:

- **Design View** - la zone de présentation visuelle principale.
- **Source View**- code d'écriture et analyse du code généré
- **Structure View**- composée de l'arbre de composant et du volet Propriété.
- **Component Tree**- montre la relation hiérarchique entre tous les composants.
- **Property Pane**- affiche les propriétés et les événements des composants sélectionnés.
- **Palette** - permet un accès rapide aux composants spécifiques à une trousse d'outils.
- **ToolBar** - permet d'accéder aux commandes couramment utilisées.
- **Context Menu** - permet d'accéder aux commandes couramment utilisées.

### IV.1.3 Installation de WindowBuilder :

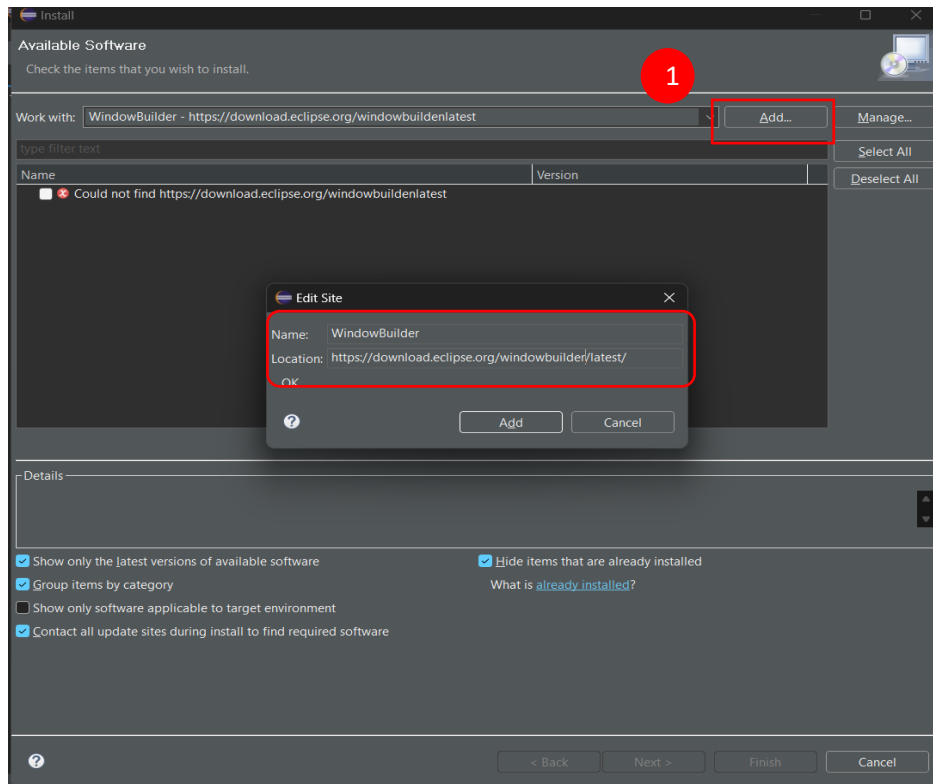
Le plugin Eclipse WindowBuilder est un concepteur Java GUI visuel, puissant et facile à utiliser permettant la création d'applications GUI Java sans vous casser la tête à écrire du code pour afficher des objets graphiques simples comme fenêtres, bouton de commandes, champs de textes... Avec WindowBuilder, vous pouvez créer des fenêtres compliquées en quelques minutes, il suffit d'utiliser le concepteur.

1. Lancer Eclipse puis Depuis le menu Help d'Eclipse, choisissez **Help > Install new Software...**

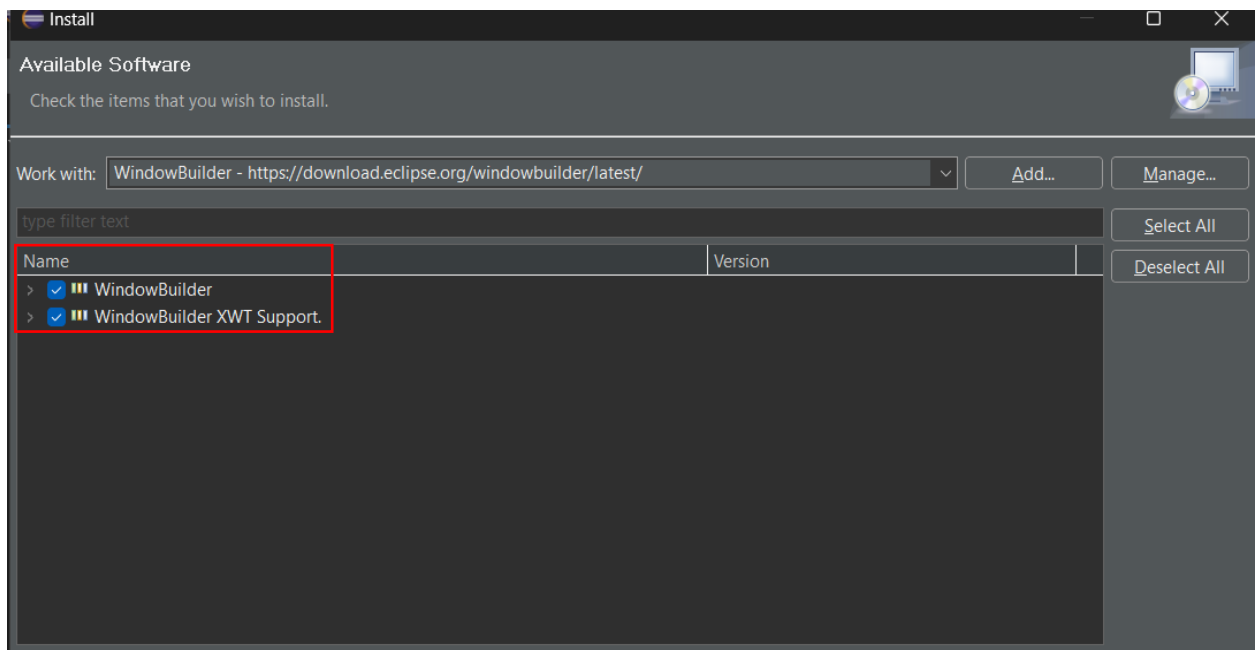


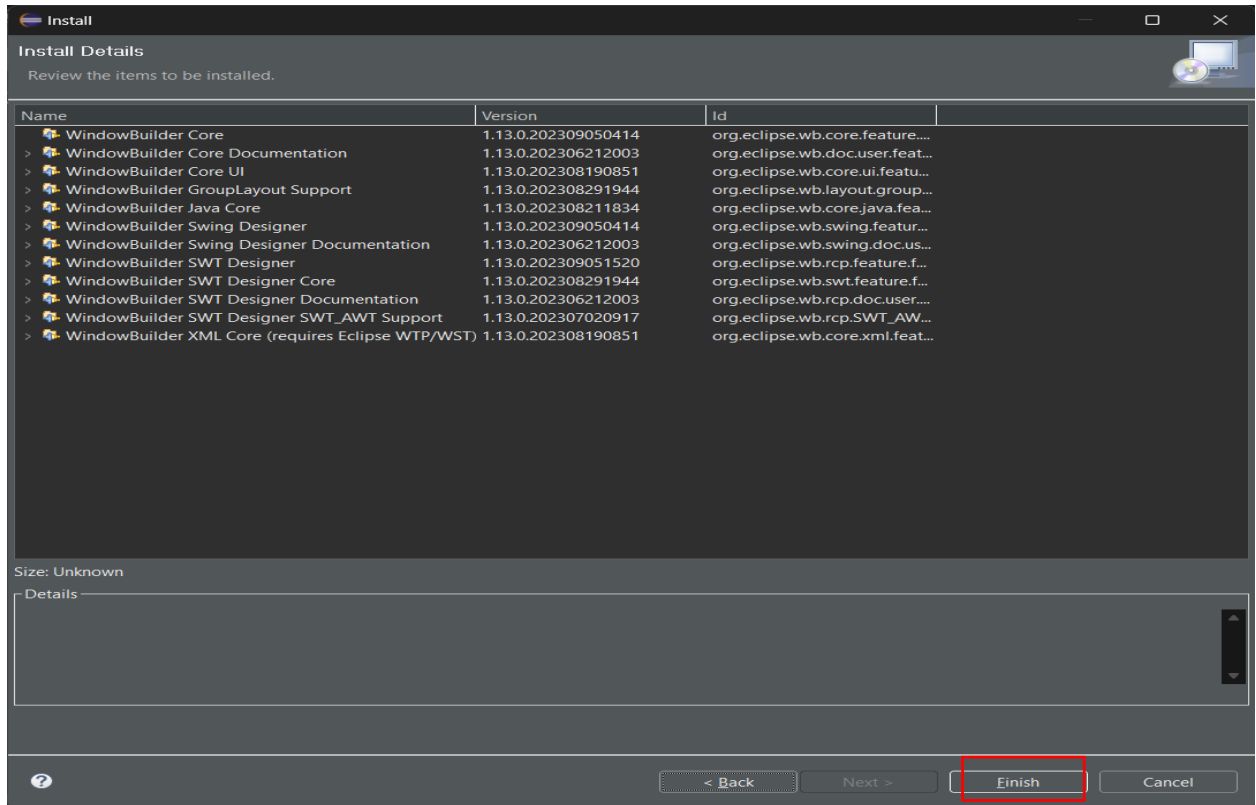
2. Dans la première fenêtre qui apparaît cliquez sur le bouton **Add** en suite, dans la boîte de dialogue apparue, dans le champ Name, saisissez un nom descriptif (comme "**WindowBuilder**") et collez le lien ci-dessus dans le champ Emplacement. Cliquez ensuite sur le bouton **OK** :

**Lien :** <https://download.eclipse.org/windowbuilder/latest/>



3. Sélectionnez toutes les **cases à cocher** qui vont apparaître, puis cliquez sur **Suivant** pour installer **WindowBuilder** :



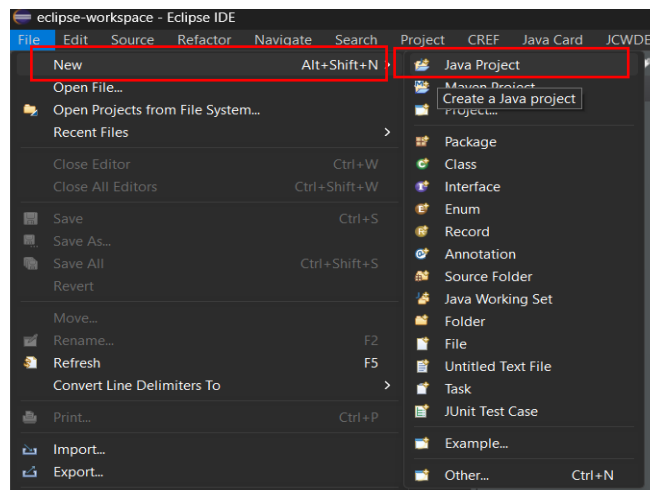


### IV.1.3 Comment utiliser WindowBuilder :

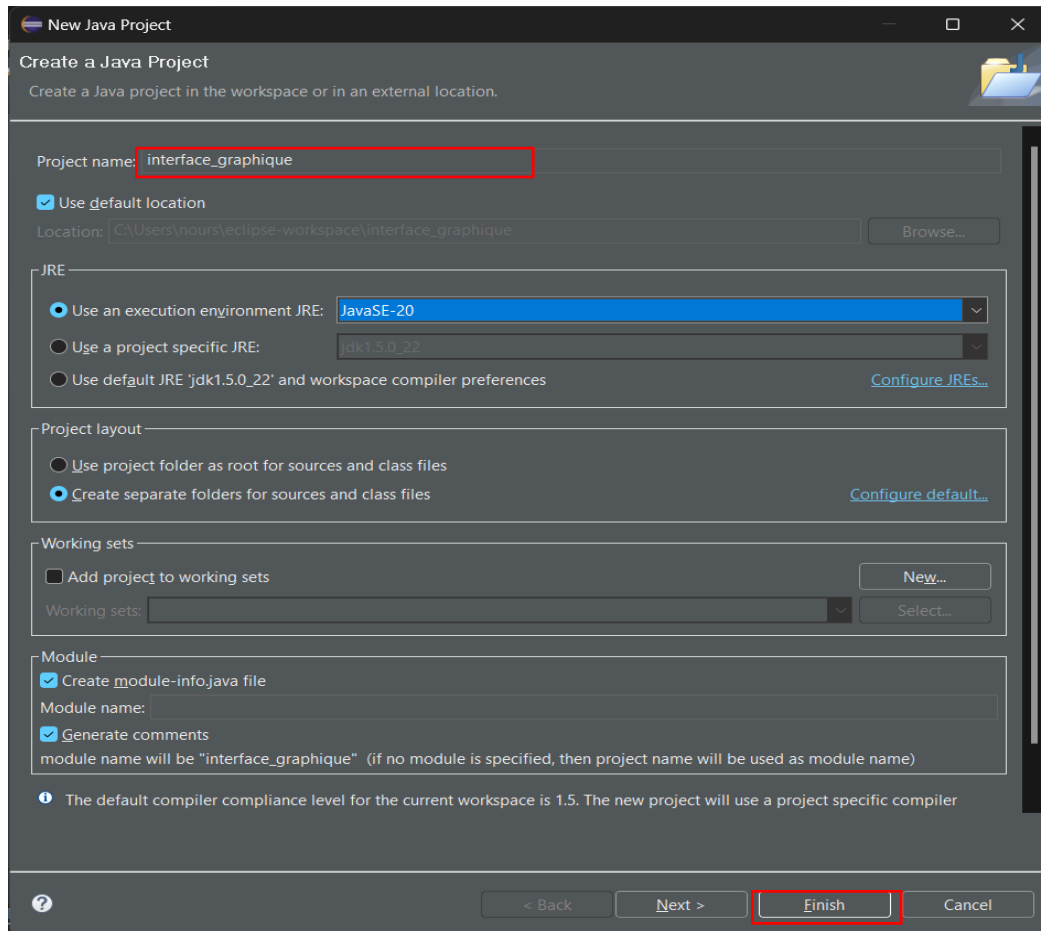
Le plugin Eclipse WindowBuilder est un concepteur

Pour utiliser WindowBuilder :

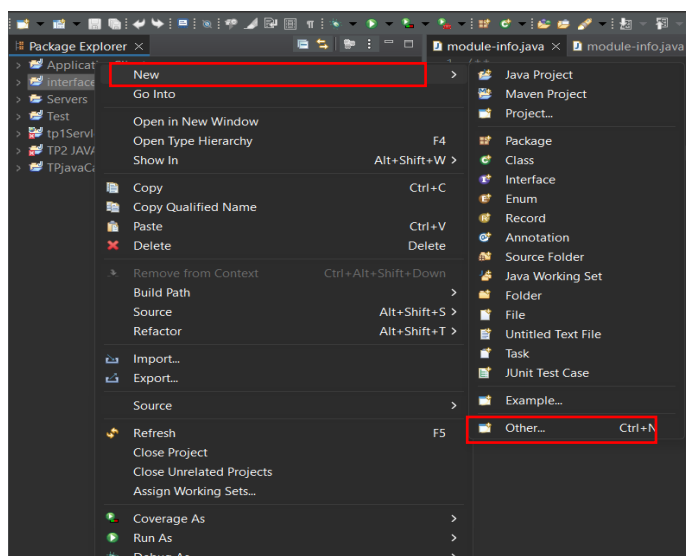
1. Créer un nouveau projet. Cliquez sur **File > New > Java Project**



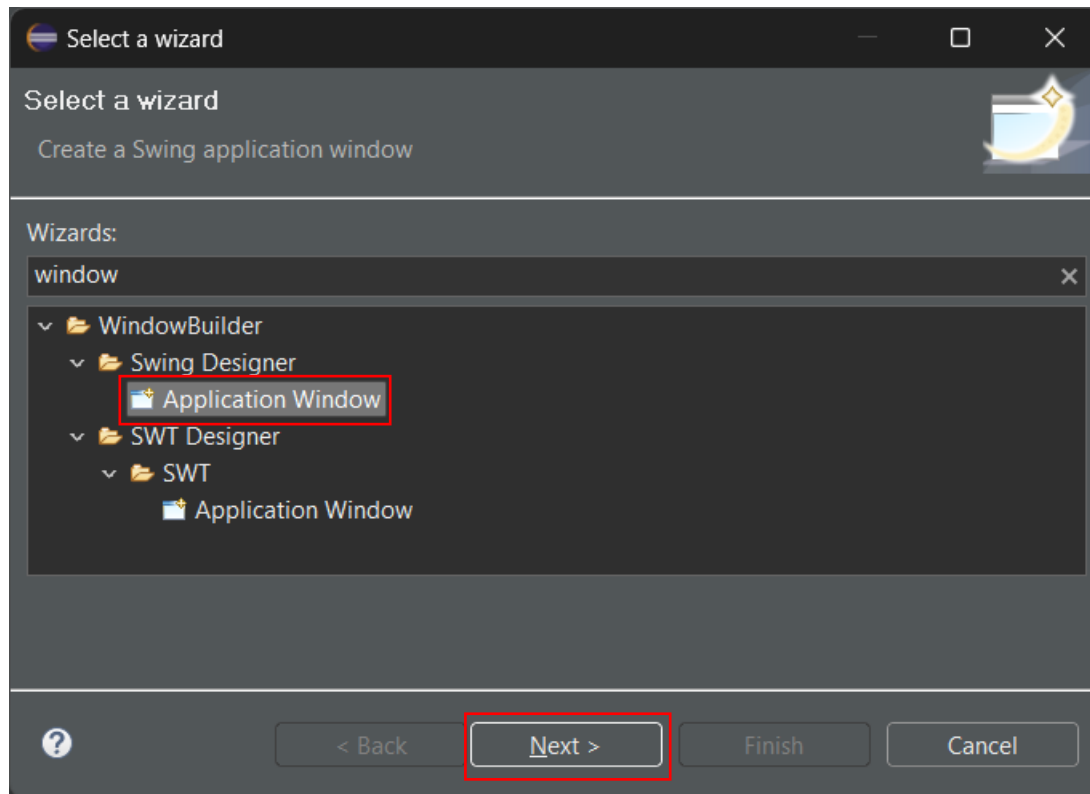
2. Saisir le nom de votre Projet (Ici **Interface\_graphique**) puis cliquer sur **Finish**



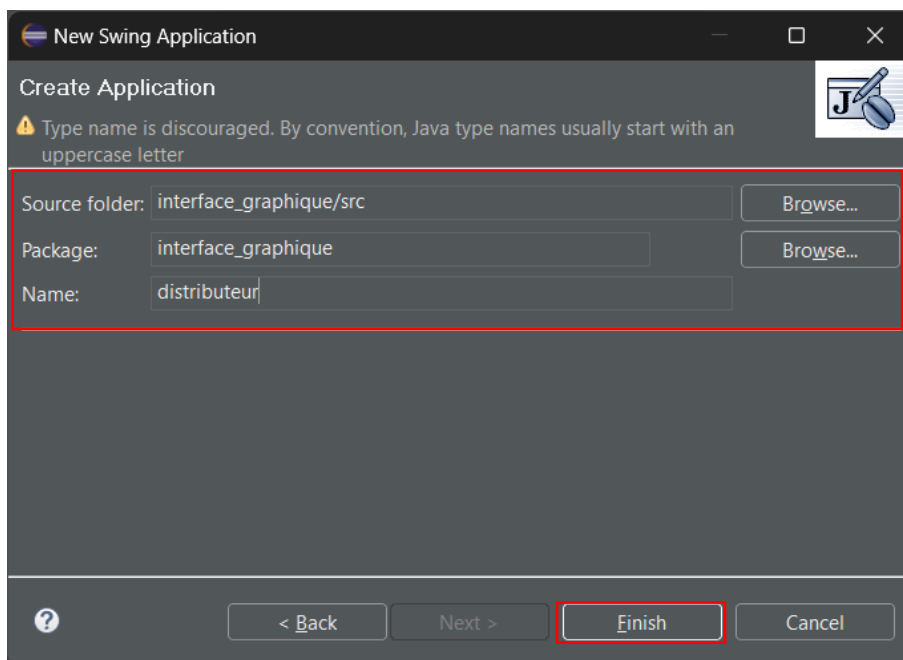
3. Après avoir créer votre Projet Java. Aller sur le dossier cliquer droit **New> Other**



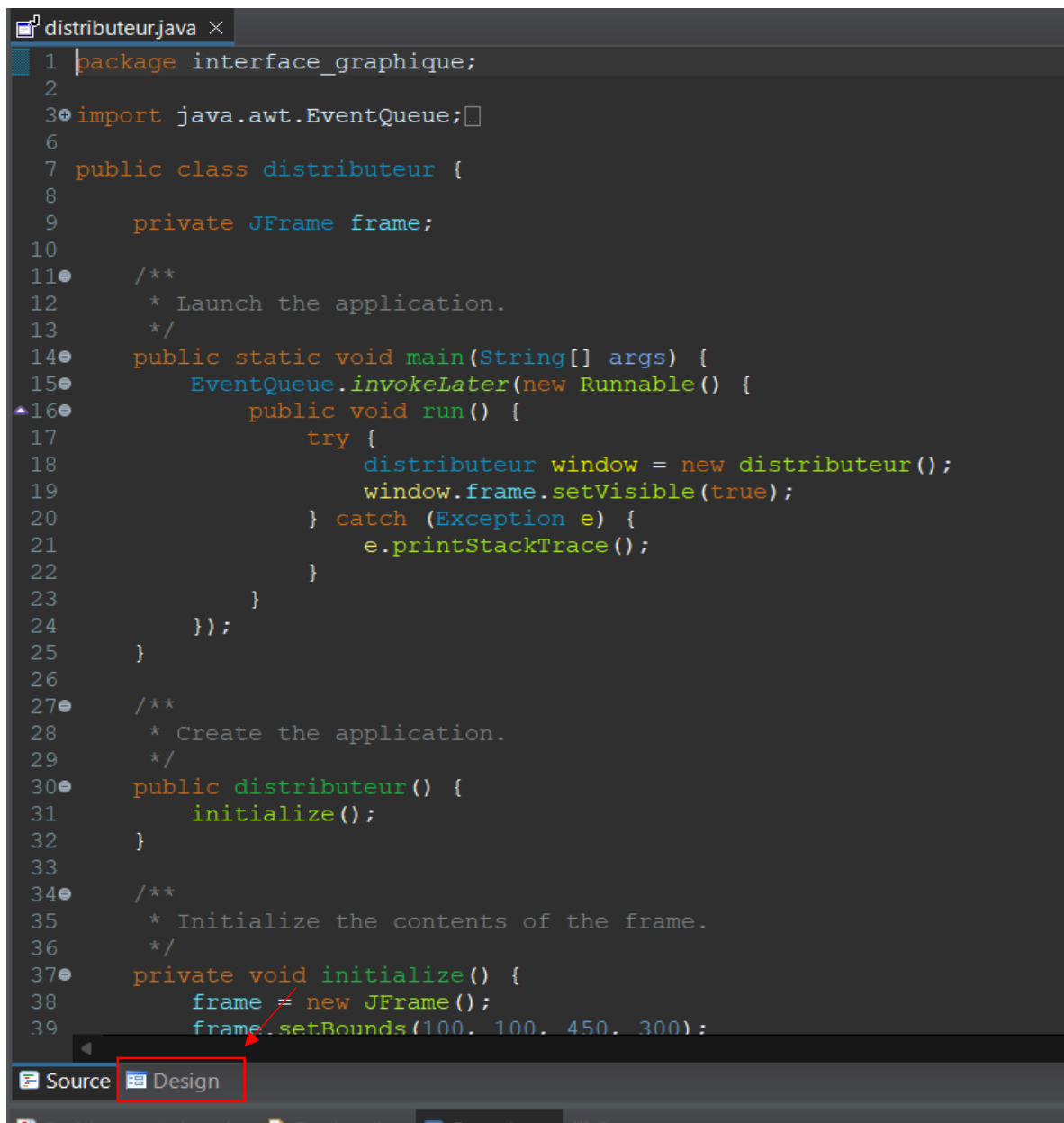
4. Choisir ensuite comme type de projet : **WindowBuilder** -> **Application Window** puis cliquer sur **Next**



5. Choisir alors le Nom de votre Projet et le package (Ici Name : **Interface\_graphique** Package **Interface\_graphique**).

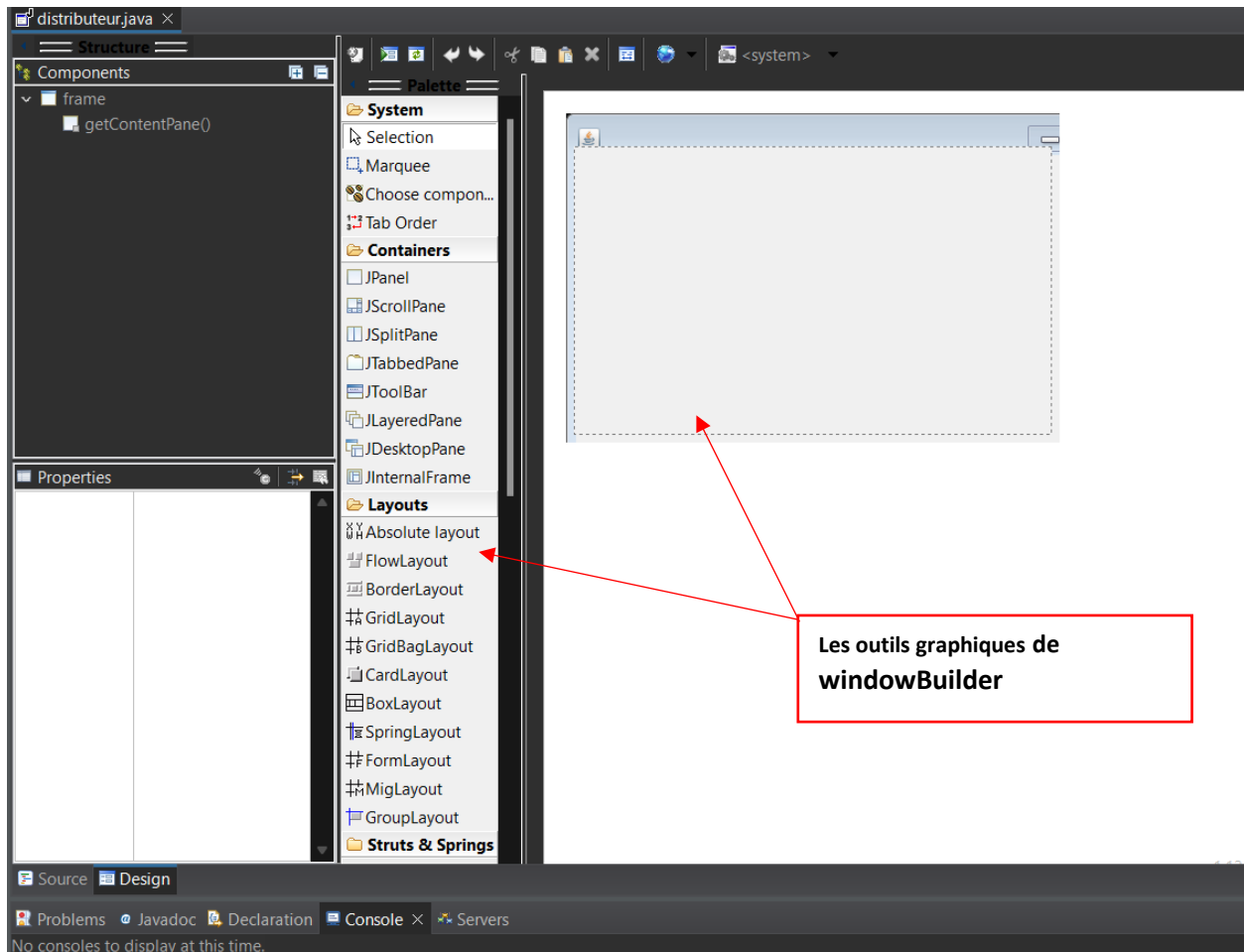


6. Après avoir cliqué sur le bouton **Finish** vous obtenez votre **projet WindowBuilder** :



```
1 package interface_graphique;
2
3 import java.awt.EventQueue;
4
5
6
7 public class distributeur {
8
9 private JFrame frame;
10
11 /**
12 * Launch the application.
13 */
14 public static void main(String[] args) {
15 EventQueue.invokeLater(new Runnable() {
16 public void run() {
17 try {
18 distributeur window = new distributeur();
19 window.frame.setVisible(true);
20 } catch (Exception e) {
21 e.printStackTrace();
22 }
23 }
24 });
25 }
26
27 /**
28 * Create the application.
29 */
30 public distributeur() {
31 initialize();
32 }
33
34 /**
35 * Initialize the contents of the frame.
36 */
37 private void initialize() {
38 frame = new JFrame();
39 frame.setBounds(100, 100, 450, 300);
```

7. Cliquez maintenant sur l'onglet **Design** pour afficher la fenêtre créée automatiquement par **WindowBuilder**, ainsi que les autres outils graphiques WYSIWYG : Containers, Layouts, Composants Swing et Awt ...



⇒ Maintenant, vous pouvez commencer à construire votre fenêtre et le code sera automatiquement générer dans la partie **code**.

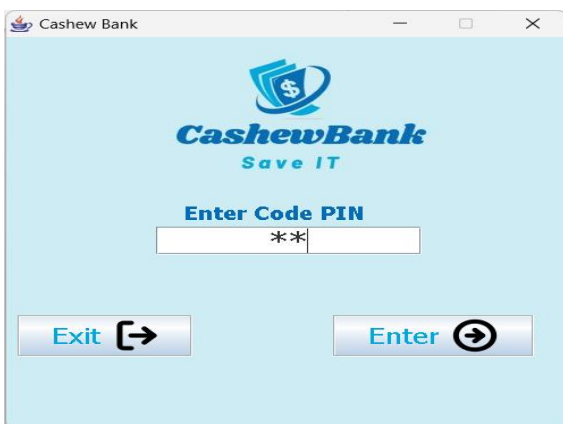


## IV.1.4 Les interfaces de notre application Client :

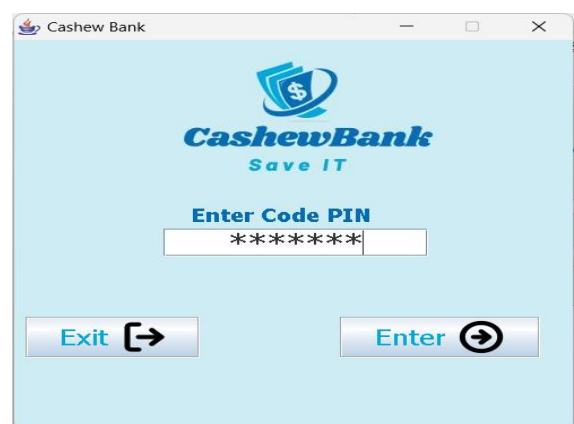
1. LA PREMIÈRE INTERFACE QUI S’AFFICHE LORS DE L’EXÉCUTION. IL FAUT SAISIR LE CODE PIN DANS LA ZONE DU TEXTE PUIS VALIDER. UN APDU SERA ENVOYER AU SERVEUR POUR CONFIRMER SI LE CODE PIN EST CORRECT OU NON.



- **1er Cas:**  
si l'utilisateur saisie un input de longueur **n'est pas égale à 4** un message d'erreur sera affiché: Wrong length code!



ou





- **2eme Cas:**

Si l'input contient un caractère alphabétique ou plus un message d'erreur s'affiche : **Wrong Input**



- **3ème cas:**

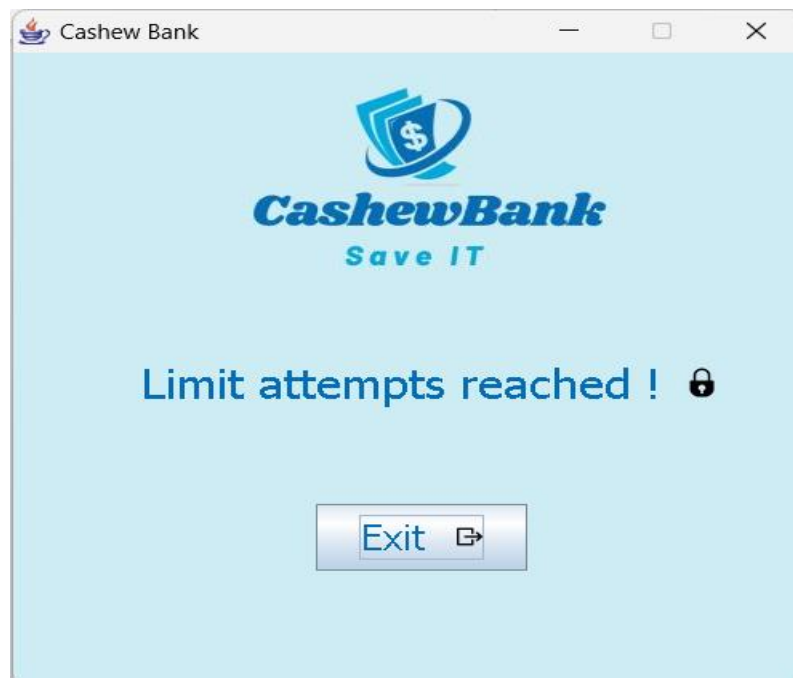
Si le code Pin saisi est **incorrect** alors une fenêtre d'erreur s'affiche: Wrong Code Pin



⇒ Vous pouvez cliquer sur **Retry** pour ressaisir votre code PIN ou bien **Exit** pour fermer le programme.

- **4ème Cas :**

Si vous avez saisi un code PIN incorrect 3 fois, votre carte sera bloquée:



- **5<sup>ème</sup> Cas :**

Si vous avez saisi le code PIN **correctement** :



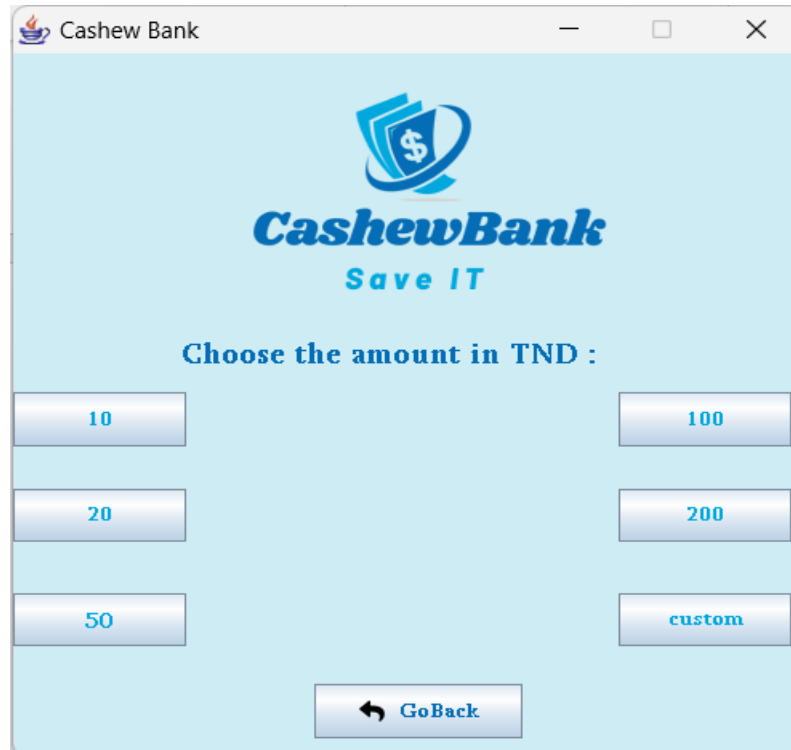
2. On se retrouve alors dans la fenêtre principale qui contient 4 boutons :

- **Check Balance** : Permet de consulter le solde .
- **Retrieve Money** : Permet de débiter un montant .
- **Cash deposit** : Permet de créditer un montant .
- **A propos** : Contient nos informations


**Check Balance :**



## Retrieve Money



Cashew Bank

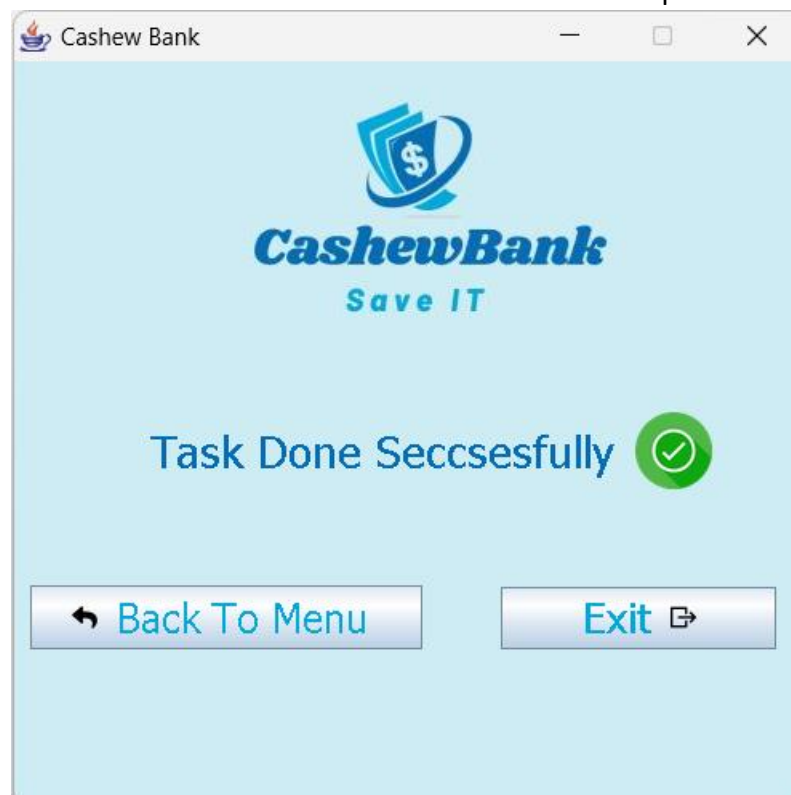
  
**CashewBank**  
Save IT

Choose the amount in TND :


|                                   |                                       |
|-----------------------------------|---------------------------------------|
| <input type="button" value="10"/> | <input type="button" value="100"/>    |
| <input type="button" value="20"/> | <input type="button" value="200"/>    |
| <input type="button" value="50"/> | <input type="button" value="custom"/> |


### 1<sup>ère</sup> Cas :

Si le montant qui vous avez choisi est valide et votre solde est suffisant pour le retrait d'argent.



Cashew Bank

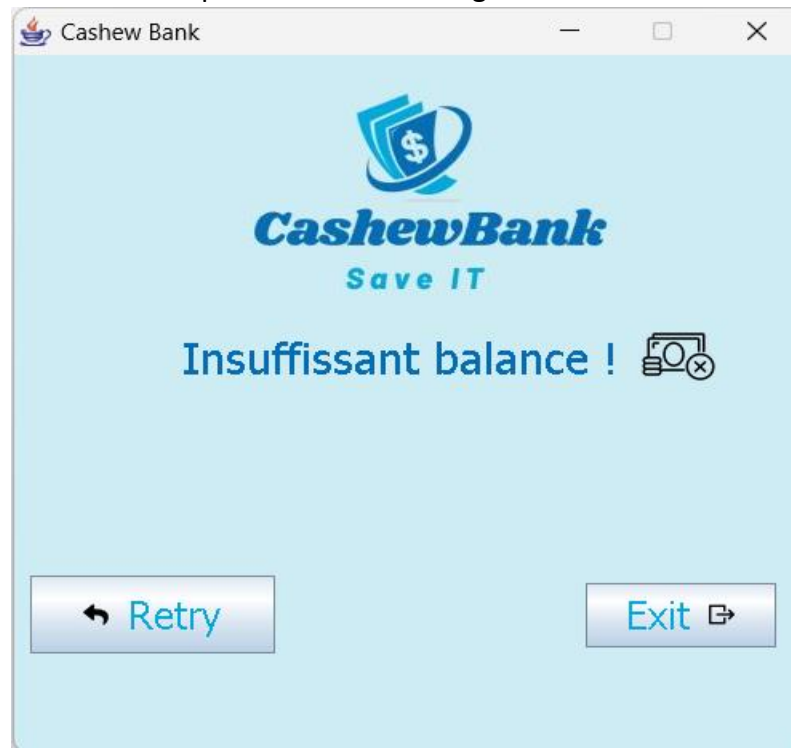
  
**CashewBank**  
Save IT

Task Done Seccsesfully 

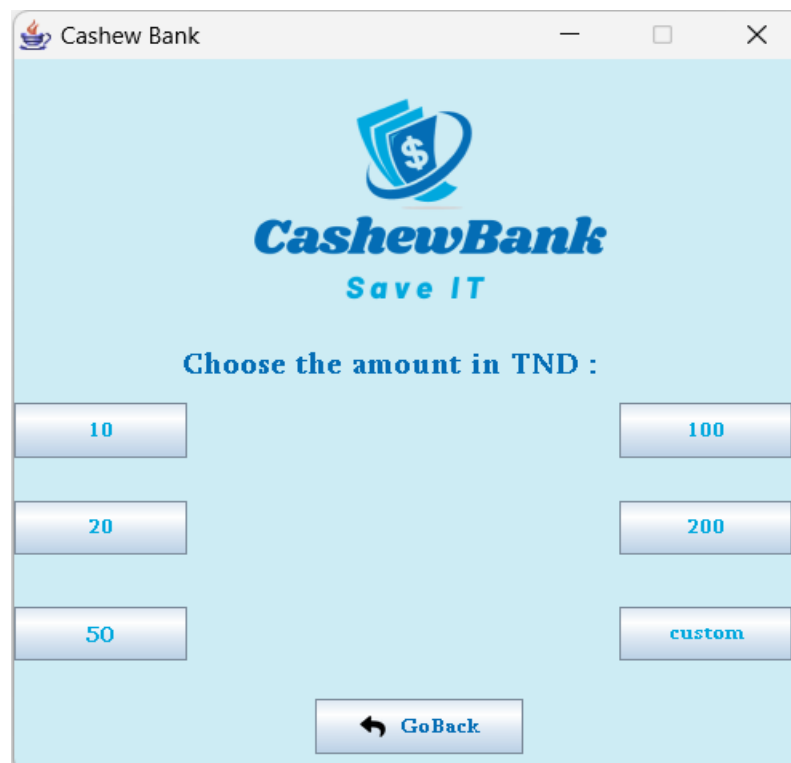
|                                               |                                       |
|-----------------------------------------------|---------------------------------------|
| <input type="button" value="↩ Back To Menu"/> | <input type="button" value="Exit ↗"/> |
|-----------------------------------------------|---------------------------------------|

## 2<sup>ème</sup> Cas :

Si vous n'avez pas le solde suffisant pour le retrait de l'argent.

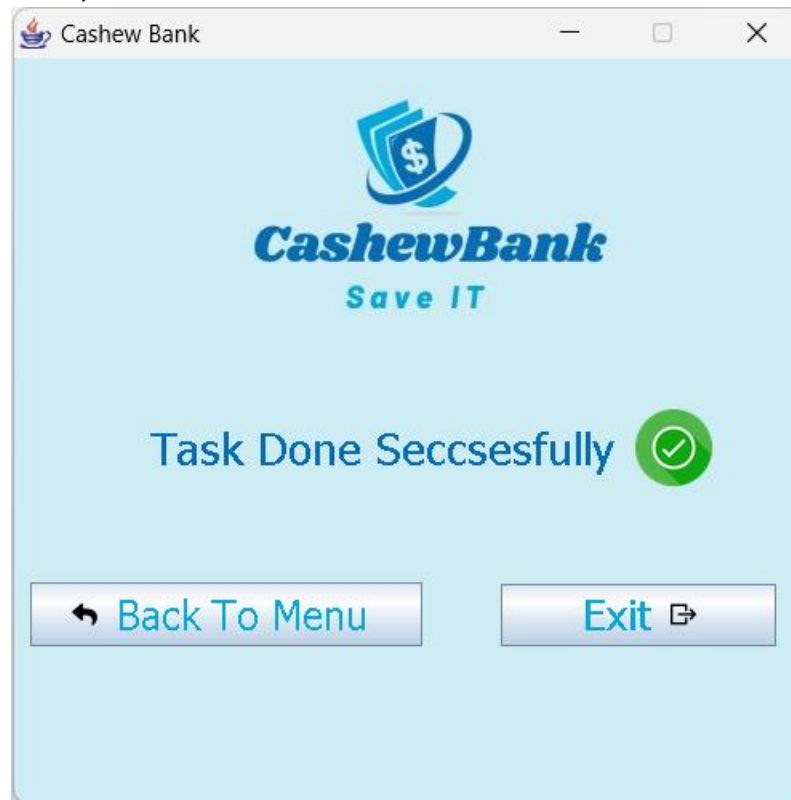


## Cash deposit



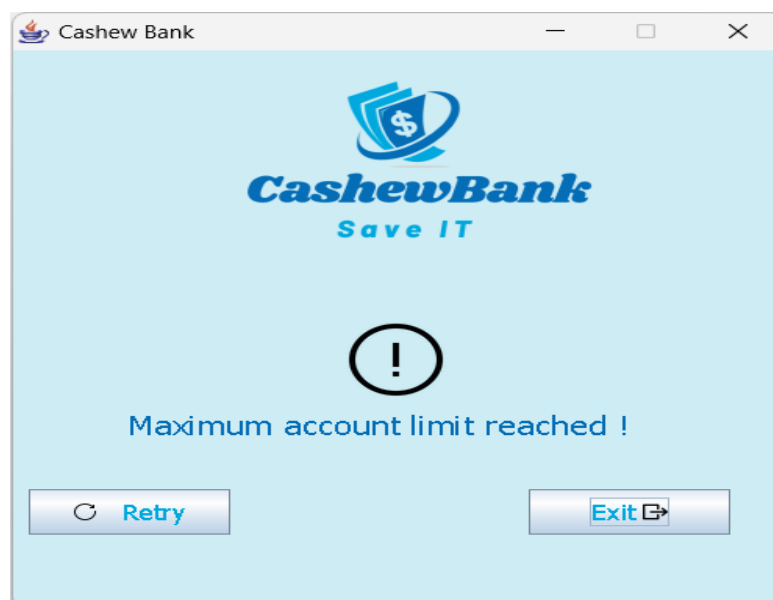
### 1<sup>ère</sup> Cas :

Si le montant qui vous avez choisi est valide et votre balance ne dépasse pas le montant autorisé (le plafond du compte :10000):

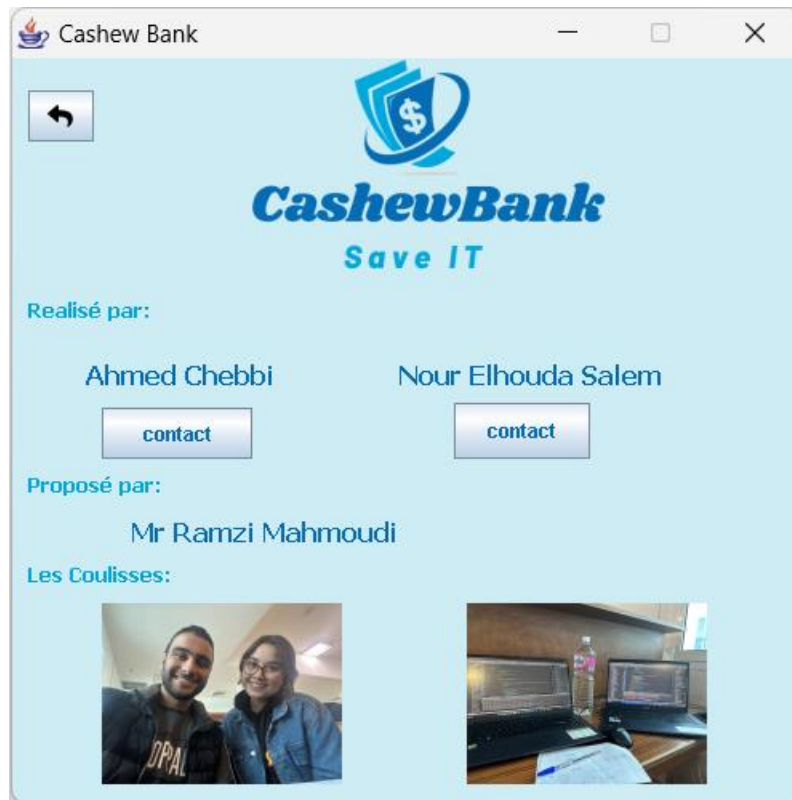


### 2<sup>ère</sup> Cas :

Si en ajoutant le montant que vous avez choisi votre balance dépasse le montant autorisé une erreur s'affiche :



## A propos



Pour plus d'information cliquer sur les boutons contact :





## IV.1.5 Implémentation des méthodes nécessaires :

### 1. Déclaration des variables :

```
static Adu apdu ;
static CadT1Client cad;
public static final byte CLA_MONAPPLET = (byte) 0xB0;
```

### 2. Implémentation de la methode Msg() :

Cette méthode permet l'envoi des Aputs.

```
public Adu Msg(byte ins, byte lc, byte[] data, byte le) throws IOException, CadTransportException{
 apdu = new Adu();
 apdu.command[Adu.CLA] = CLA_MONAPPLET;
 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);
 System.out.println(apdu);
 return apdu;
}
```

### 3. Implémentation de la methode Connect() :

Cette méthode permet la connexion avec la carte.

```
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;
 }
}
```

## 4.Implémentation de la methode Select() :

Cette méthode permet la sélection de notre applet dans la carte.

```
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);
 }

}
```

## 5. Implémentation de la methode Deselect() :

```
public void Deselect(){
 /* 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;
 }
}
```

## IV.2.Partie Serveur (Back-End) :

### 1.Declaration des variables et des constantes :

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

public static final byte INS_Retrive_10 = 0x01;
public static final byte INS_Retrive_20 = 0x02;
public static final byte INS_Retrive_50 = 0x03;
public static final byte INS_Retrive_100 = 0x04;
public static final byte INS_Retrive_200 = 0x05;
public static final byte INS_Retrive_Costume = 0x06;

public static final byte INS_add_10 = 0x11;
public static final byte INS_add_20 = 0x12;
public static final byte INS_add_50 = 0x13;
public static final byte INS_add_100 = 0x14;
public static final byte INS_add_200 = 0x15;
public static final byte INS_add_Costume = 0x16;

public static final byte INS_Show_Balance = 0x20;

public static final byte INS_VIRIF_PIN = 0x40;
private final static byte PIN_SIZE =4;
private final static byte PIN_TRY_LIMIT=3;
|
private OwnerPIN pin;
private byte [] balance;
```

### 2.Implementations des méthodes nécessaires :

#### 2.1 constructeur :

```
private Bank() {
 // Initialize PIN with maximum tries and PIN size
 pin = new OwnerPIN(PIN_TRY_LIMIT, PIN_SIZE);
 // Set the initial PIN value (example: 1234)
 byte[] initialPIN = {(byte)0x01, (byte)0x02, (byte)0x03, (byte)0x04};
 pin.update(initialPIN, (short)0, PIN_SIZE);

 balance= new byte [] {(byte)0x00, (byte)0x00, (byte)0x01, (byte)0x00, (byte)0x00};
}
}
```

#### 2.2 Implementation de la methode install() :

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

### 2.3 Implementation de la methode show\_balance() :

```
public void show_balance(APDU apdu)
{
 byte[] buffer = apdu.getBuffer();
 for (int i=0 ; i<balance.length ; i++)
 {
 buffer[i] = balance[i];
 }
 apdu.setOutgoingAndSend((short) 0, (short) 5);
}
```

### 2.4 Implementation de la methode check() :

```
public boolean check(byte[] pinAttempt, short offset, byte length)
 throws ArrayIndexOutOfBoundsException, NullPointerException {
 // TODO Auto-generated method stub
 return pin.check(pinAttempt, offset, length);
}
```

### 2.5 rédefintion de la methode 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_VIRIF_PIN:
 PIN_Code(apdu);
 break;
 case INS_Retrive_Costume:
 Retrive_Costume(apdu);
 break;
 case INS_Retrive_10 :
 Retrive_10(apdu);
 break;
 case INS_Retrive_20 :
 Retrive_20(apdu);
 break;
 case INS_Retrive_50 :
 Retrive_50(apdu);
 break;
 case INS_Retrive_100 :
 Retrive_100(apdu);
 break;
 case INS_Retrive_200 :
 Retrive_200(apdu);
 break;
 case INS_add_10 :
 add_10(apdu);
 }
}
```

## 2.6 Implementation de la methode add() :

```
public static byte[] add(byte[] array1, byte[] array2) {
 byte[] r = new byte[array1.length]; // assuming both arrays are same length
 int carry = 0;
 for (int i = array1.length - 1; i >= 0; i--) { // LSB to MSB

 int sum = (array1[i])+(array2[i])+carry;
 //System.out.println(sum);
 r[i] = (byte) (sum %10);
 carry = (int) sum / 10;
 }
 return r;
}
```

## 2.7 implementation de la methode subtract() :

```
public static byte[] subtract(byte[] num1, byte[] num2) {

 byte[] result = new byte[num1.length];
 int borrow = 0;

 for (int i = num1.length - 1; i >= 0; i--) {
 // Subtracting current digits along with borrow
 int temp = (num1[i] & 0xFF) - (num2[i] & 0xFF) - borrow;

 // If the subtraction is negative, add 10 and set borrow
 if (temp < 0) {
 temp += 10;
 borrow = 1;
 } else {
 borrow = 0;
 }

 result[i] = (byte) temp;
 }

 return result;
}
```

## 2.8 Rédéfinition de la methode PIN\_CODE() :

```
public void PIN_Code(APDU apdu) {
 byte[] buffer = apdu.getBuffer();
 short numBytes = apdu.setIncomingAndReceive();
 if (numBytes != PIN_SIZE) {
 buffer[0] = pin.getTriesRemaining();
 apdu.setOutgoingAndSend((short) 0, (short) 1);
 ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
 }
 if (!check(buffer, ISO7816.OFFSET_CDATA, PIN_SIZE)) {
 buffer[0] = pin.getTriesRemaining();
 apdu.setOutgoingAndSend((short) 0, (short) 1);
 ISOException.throwIt(ISO7816.SW_SECURITY_STATUS_NOT_SATISFIED);
 }
 else
 {
 buffer[0] = pin.getTriesRemaining();
 apdu.setOutgoingAndSend((short) 0, (short) 1);}
}
```