

REPUBLIQUE DU CAMEROUN

\*\*\*\*\*

MINISTRE DE  
L'ENSEIGNEMENT SUPERIEUR

\*\*\*\*\*

UNIVERSITE DE NGAOUNDERE

\*\*\*\*\*

FACULTE DES SCIENCES

\*\*\*\*\*

DEPARTEMENT DE MATH-INFO

\*\*\*\*\*



REPUBLIC OF CAMEROON

\*\*\*\*\*

MINISTRY OF HIGHER  
EDUCATION

\*\*\*\*\*

UNIVERSITY OF  
NGAOUNDERE

\*\*\*\*\*

FACULTY OF SCIENCES

\*\*\*\*\*

DEPARTMENT OF MATH-  
INFO

\*\*\*\*\*

UNITE D'ENSEIGNEMENT : SYSTEMES D'EXPLOITATION MOBILES (INF 345)

TRAVAUX PRATIQUES LICENCE 3 INFORMATIQUE

MEMBRES DU GROUPE :

1-KABIRROU HAMADOU TIZI 16B093FS

2-NENBA JONATHAN

3-VENCESLAS MINAOU

4-KOTVA GOUDOUNGOU SAMUEL 16B119FS

5-GONG-MOGA EMMANUEL GAISSALA 16B161FS

6-BABA SOULEY HAMIDOU 16A987FS

7-MAHAMAT II AHMAT 16B126FS

8-AMONO AYMAR

9-SIBEU DOMGUE AROLE LESLY 16B021FS

10-MBOUCHE BOMDA ULRICHE

11-OUENA DJERAKOULA FABRICE 16B427FS

12-NDIANAIBEYE NGARLENAN 15B280FS

ENSEIGNANT : *Dr.-Ing.* FRANKLIN TCHAKOUNTE

Année Académique 2018/2019

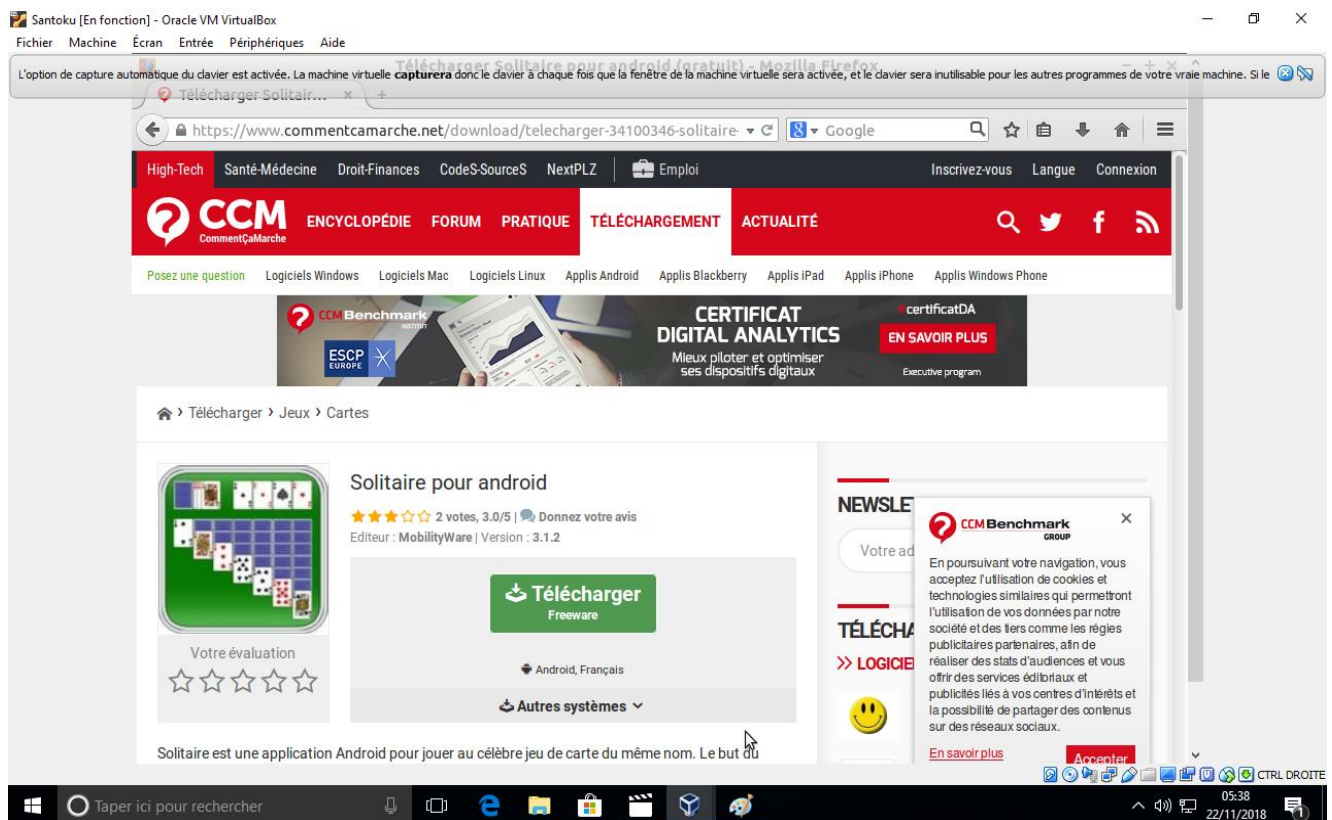
## SUJET 1 :

- 1°) Télécharger une application de votre choix.
- 2°) La décompiler.
- 3°) Dissequer le contenu du fichier manifeste et y ressortir les différentes composantes telles que vues en cours.

## REPONSES

### 1°) TELECHARGEMENT

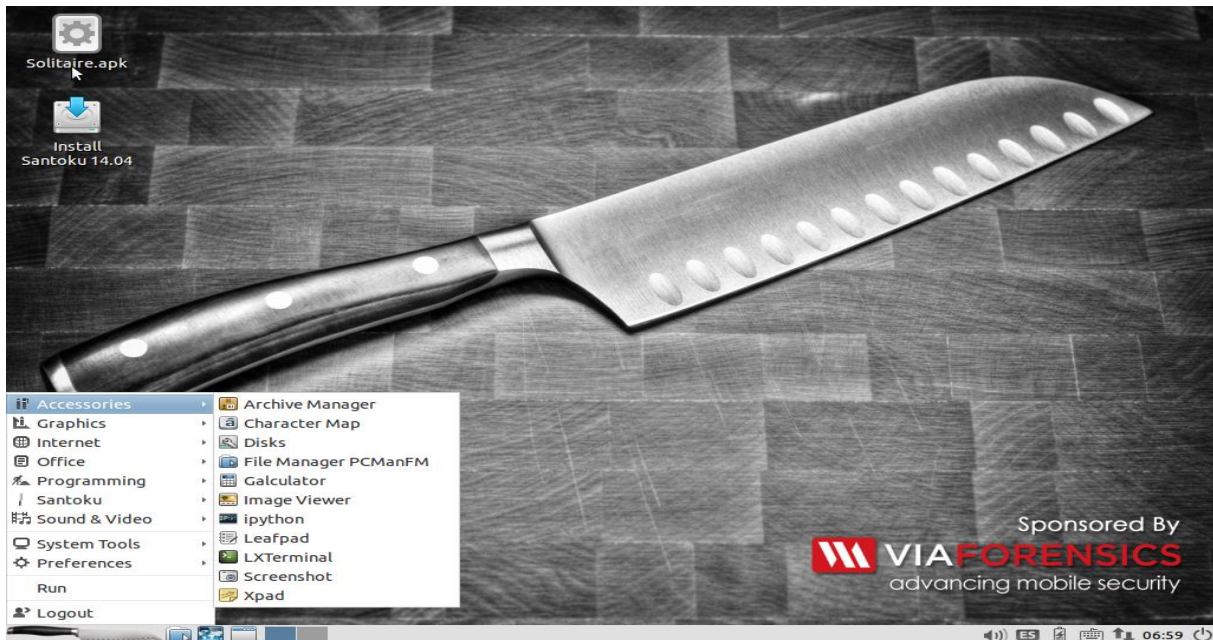
L'application que nous avons choisie est Solitaire pour Android qui est une application permettant le jeu des cartes. Nous la téléchargeons ainsi que le montre la figure suivante :



### 2°) DECOMPILATION

Le compilateur utilisé ici est le Java Decompiler déjà intégré à SANTOKU. Cette décompilation se fait en plusieurs étapes suivantes :

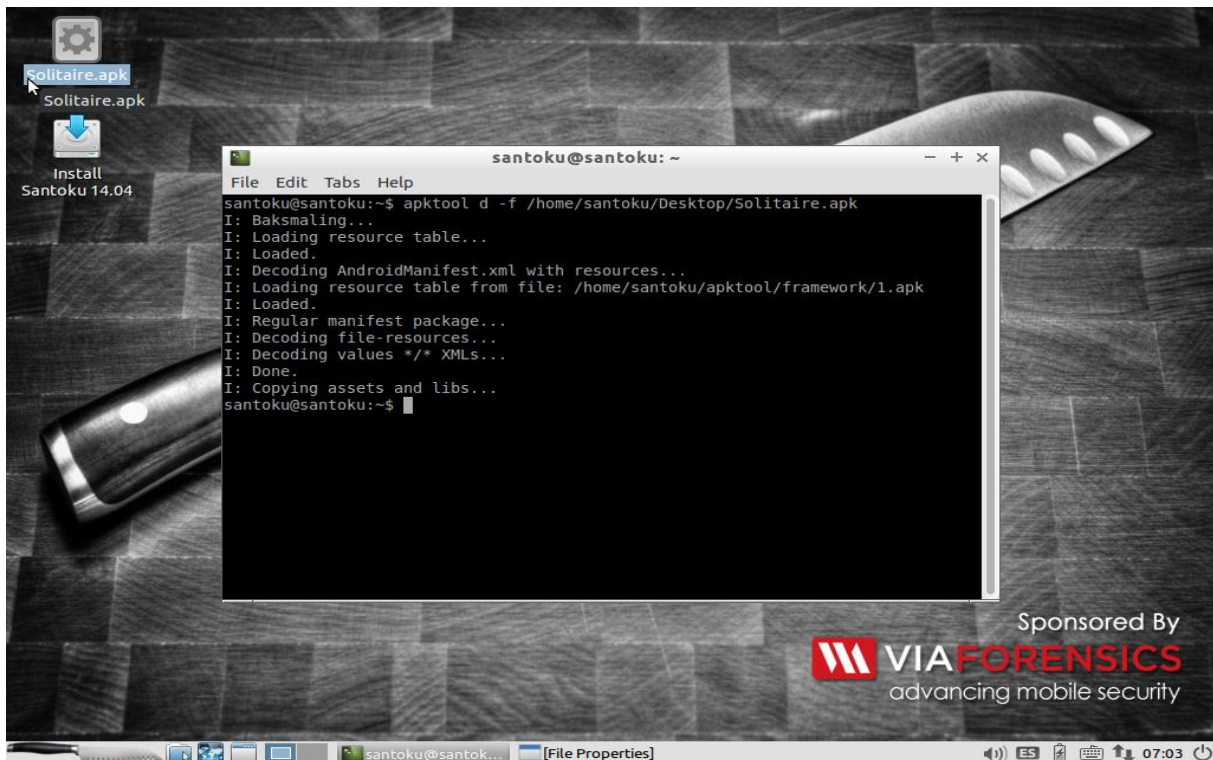
1<sup>re</sup> étape : aller sur le couteau, ensuite sur accessoires, cliquer sur LX Terminal telle que le montre la figure ci-dessous :



2eme étape : ouvrir le LX Terminal, taper la commande :

« apktool d -f /home/santoku/Desktop/Solitaire.apk »

Le processus de décompilation s'affiche librement telle que l'indique la figure suivante :



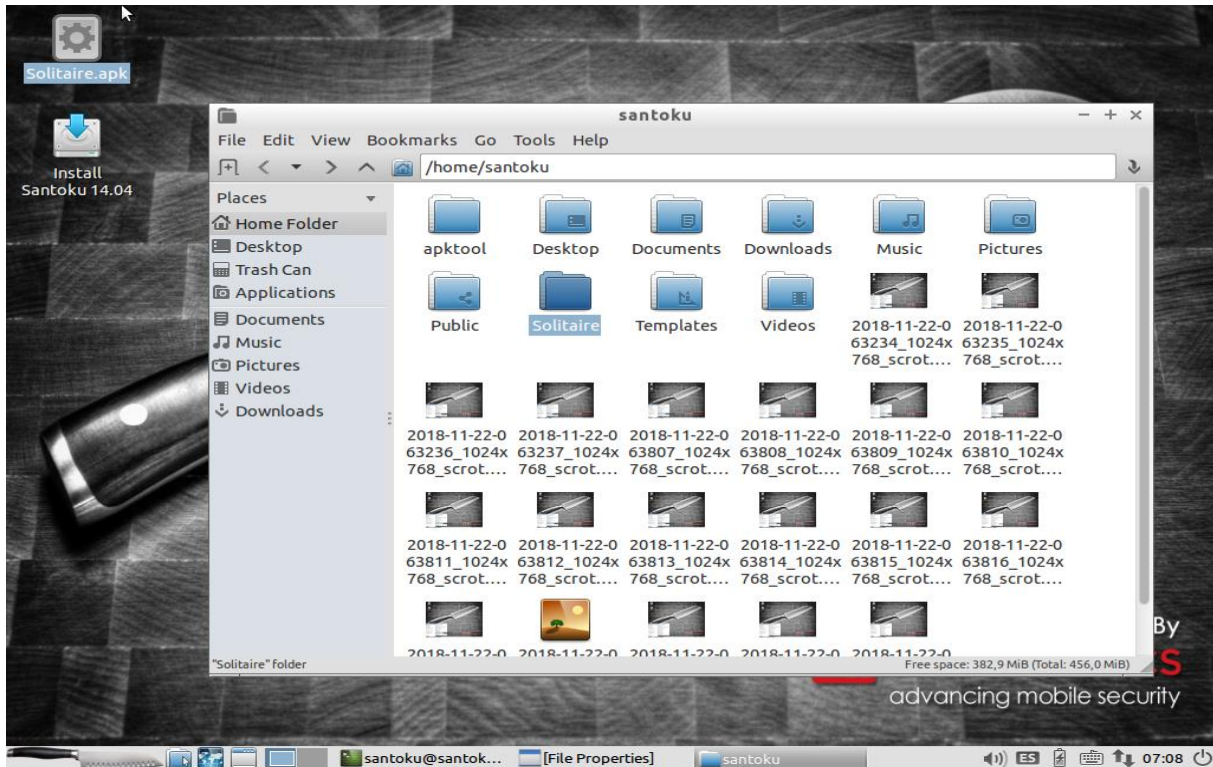
Ainsi la décompilation est terminée.



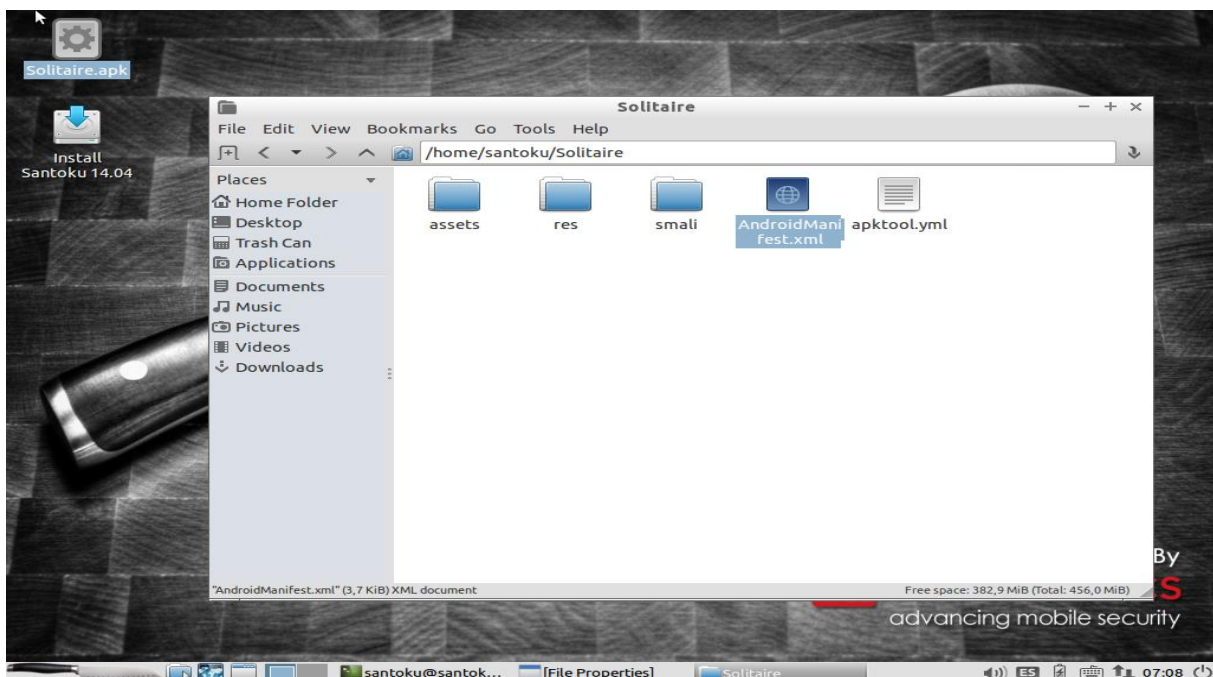
### 3°) FICHER MANIFESTE ET COMPOSANTES

Le fichier manifeste est celui la qui permet la configuration de l'application.

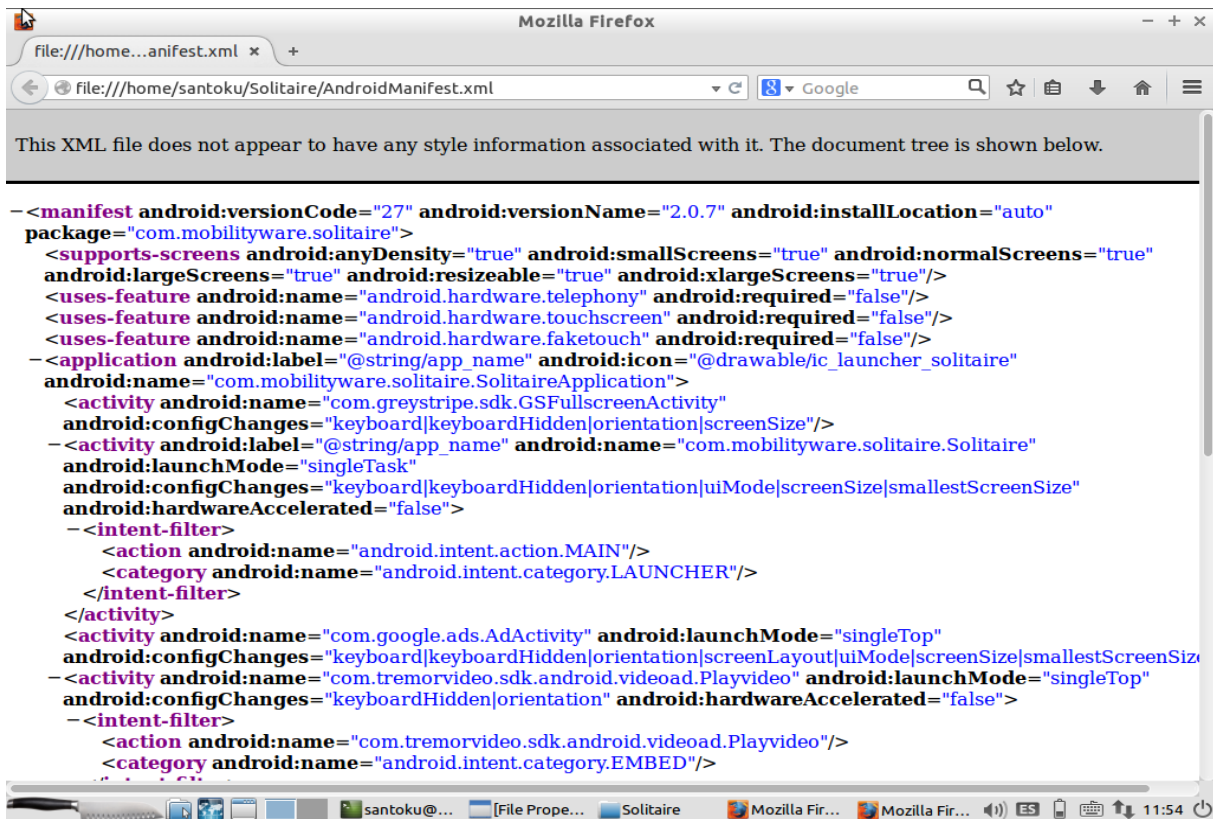
Pour ouvrir le fichier manifeste, nous allons dans file manager se trouvant juste à droite du couteau, on clique sur Solitaire ;la figure suivante illustre cela :



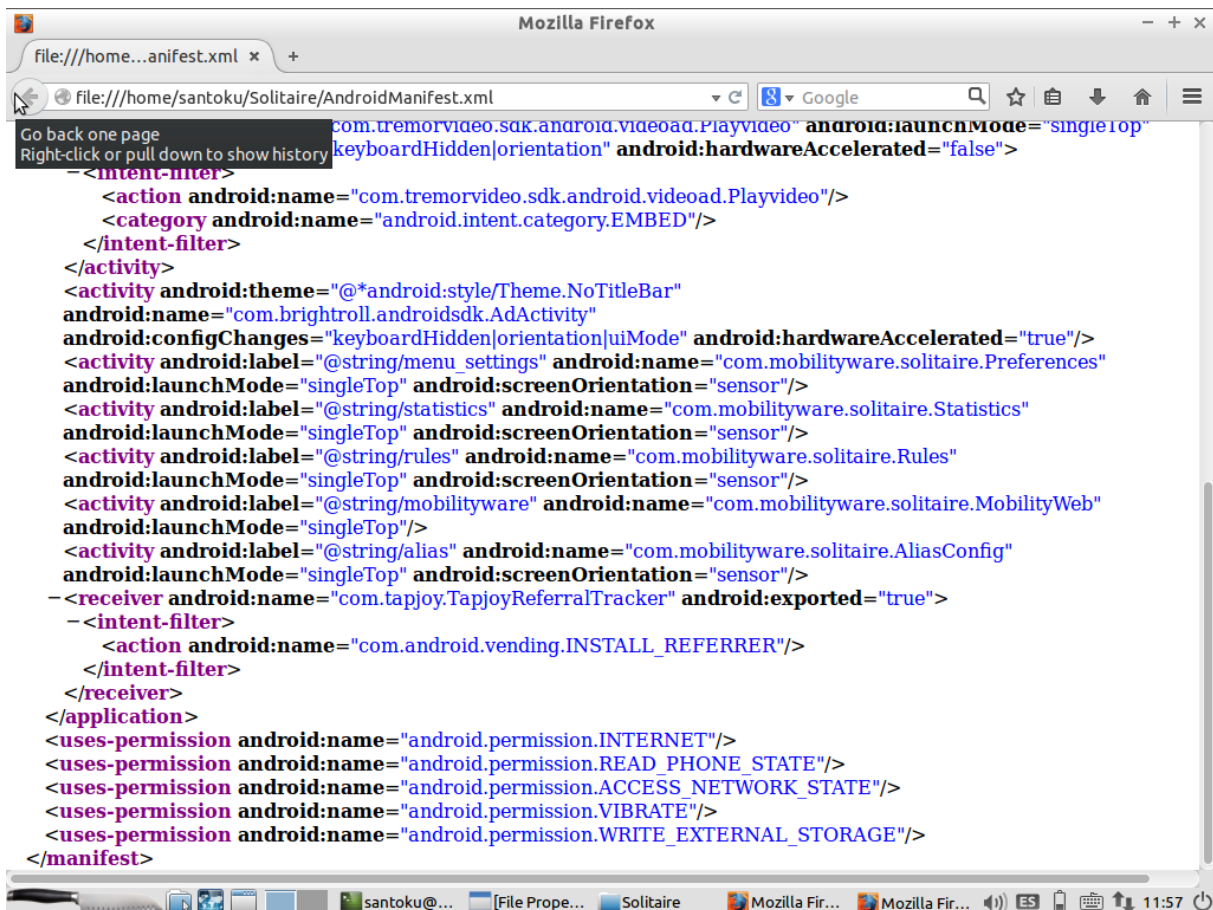
Une page s'affiche et l'on clique sur AndroidManifest.xml comme l'indique la figure suivante :



Nous ouvrons donc notre fichier manifeste et il apparait comme ceci :



```
-<manifest android:versionCode="27" android:versionName="2.0.7" android:installLocation="auto"
package="com.mobilityware.solitaire">
  <supports-screens android:anyDensity="true" android:smallScreens="true" android:normalScreens="true"
  android:largeScreens="true" android:resizeable="true" android:xlargeScreens="true"/>
  <uses-feature android:name="android.hardware.telephony" android:required="false"/>
  <uses-feature android:name="android.hardware.touchscreen" android:required="false"/>
  <uses-feature android:name="android.hardware.faketouch" android:required="false"/>
  <application android:label="@string/app_name" android:icon="@drawable/ic_launcher_solitaire"
  android:name="com.mobilityware.solitaire.SolitaireApplication">
    <activity android:name="com.greystripe.sdk.GSFullscreenActivity"
    android:configChanges="keyboard|keyboardHidden|orientation|screenSize"/>
    <activity android:label="@string/app_name" android:name="com.mobilityware.solitaire.Solitaire"
    android:launchMode="singleTask"
    android:configChanges="keyboard|keyboardHidden|orientation|uiMode|screenSize|smallestScreenSize"
    android:hardwareAccelerated="false">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
    <activity android:name="com.google.ads.AdActivity" android:launchMode="singleTop"
    android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"/>
    <activity android:name="com.tremorvideo.sdk.android.videoad.Playvideo" android:launchMode="singleTop"
    android:configChanges="keyboardHidden|orientation" android:hardwareAccelerated="false">
      <intent-filter>
        <action android:name="com.tremorvideo.sdk.android.videoad.Playvideo"/>
        <category android:name="android.intent.category.EMBED"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
```



```
com.tremorvideo.sdk.android.videoad.Playvideo" android:launchMode="singleTop"
keyboardHidden|orientation" android:hardwareAccelerated="false">
  <intent-filter>
    <action android:name="com.tremorvideo.sdk.android.videoad.Playvideo"/>
    <category android:name="android.intent.category.EMBED"/>
  </intent-filter>
</activity>
<activity android:theme="@*android:style/Theme.NoTitleBar"
android:name="com.brightroll.android.sdk.AdActivity"
android:configChanges="keyboardHidden|orientation|uiMode" android:hardwareAccelerated="true"/>
<activity android:label="@string/menu_settings" android:name="com.mobilityware.solitaire.Preferences"
android:launchMode="singleTop" android:screenOrientation="sensor"/>
<activity android:label="@string/statistics" android:name="com.mobilityware.solitaire.Statistics"
android:launchMode="singleTop" android:screenOrientation="sensor"/>
<activity android:label="@string/rules" android:name="com.mobilityware.solitaire.Rules"
android:launchMode="singleTop" android:screenOrientation="sensor"/>
<activity android:label="@string/mobilityware" android:name="com.mobilityware.solitaire.MobilityWeb"
android:launchMode="singleTop"/>
<activity android:label="@string/alias" android:name="com.mobilityware.solitaire.AliasConfig"
android:launchMode="singleTop" android:screenOrientation="sensor"/>
<receiver android:name="com.tapjoy.TapjoyReferralTracker" android:exported="true">
  <intent-filter>
    <action android:name="com.android.vending.INSTALL_REFERRER"/>
  </intent-filter>
</receiver>
</application>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
</manifest>
```

Il contient plusieurs composantes à savoir :

- a) **Les activités** qui sont des bouts de code qui gèrent l'interface graphique de l'application. Parmi elles, pour notre application, nous avons :
  - La première : chargée de gérer la taille de l'écran et les entrées (clavier et orientations) sur celui-ci ;
  - La deuxième : contenant le MAIN et de ce fait, c'est l'activité principale car elle contient aussi les intents de communication pour l'amorçage d'autre activités (présence du LAUNCHER) ;
  - Comme activités 'particulières' si on peut le dire, nous avons les **ACTIONS** qui sont les opérations possibles sur une activité déclarée. Pour la quatrième Activité par exemple, on a l'action qui permet à notre application de lire une vidéo par le bout de code suivant :

```
<action android:name="com.tremorvideo.sdk.android.Videoad.Playvideo"/>
```

- b) **Les Broadcast Receiver** qui représentent liens pour les échanges entre les ressources nécessaires à l'exécution d'une action dans une activité. Nous en avons un (1) seul pour notre application dont l'utilité est liée à la communication pour les actions d'installation à travers les chemins absolus.
- c) **Les permissions** qui sont les différentes autorisations systèmes demandées par notre application à un certain moment de son utilisation. Nous avons par exemple :
  - La permission pour se connecter à INTERNET ;
  - La permission de déclencher une Vibration ;
  - La permission d'écrire sur support de stockage Externe ;

Les broadcast receiver à travers les récepteurs de diffusion et le gestionnaire du contenu.

Nous remarquons que ce manifeste déclare cinq types de permissions ; et du fait de l'usage des permissions l'application est donc capable d'utiliser les features protégés.

## SUJET 2 :

- 1) Télécharger une application Android de votre choix.
- 2) La décompiler.
- 3) Retrouver le code source et les classes de cette application.
- 4) La recompiler pour avoir l'application initiale

## REPONSES

### 1. Télécharger une application Android de votre choix

Une application ANDROID c'est un programme exécutable sur un appareil mobile. Lorsqu'on la télécharge, on télécharge en réalité un fichier qui va être installé puis exécuté par le système d'exploitation de notre mobile. Ce fichier est codé dans un langage de **développement mobile** spécifique à notre appareil.

L'application que nous avons choisie est : **VLC media player** qui est le **lecteur vidéo et audio gratuit et open source** le plus répandu, idéal pour jouer tout format et remplacer le lecteur média de votre smartphone. Nous l'avons téléchargé ainsi que le montre la figure suivante :

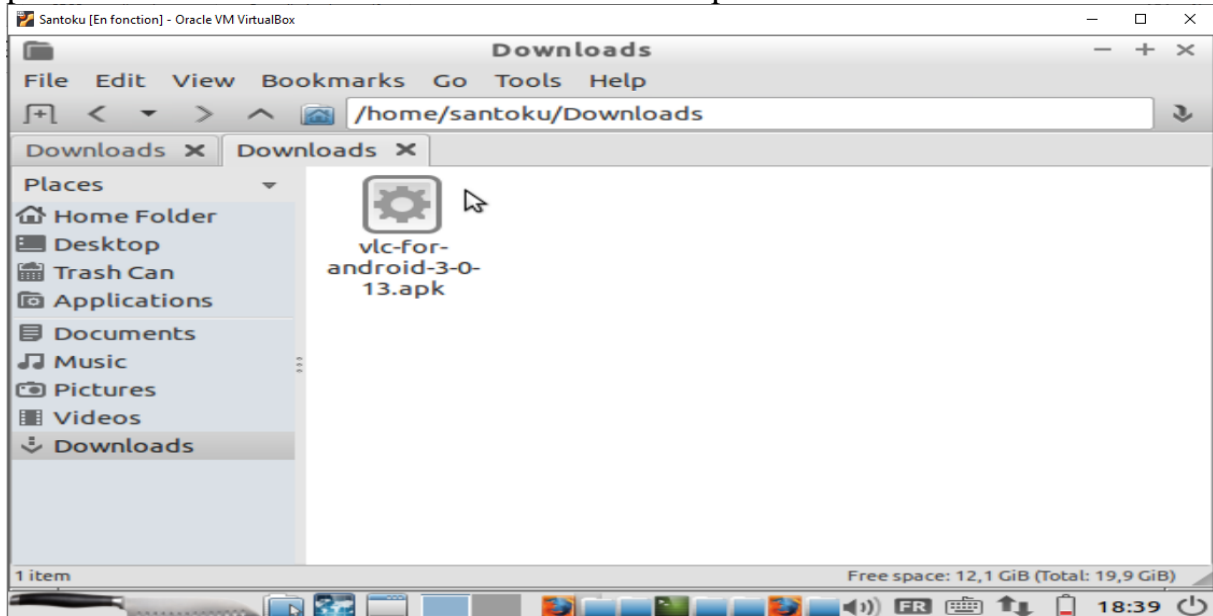


### 2. Décompilation, code source et les classes de cette application

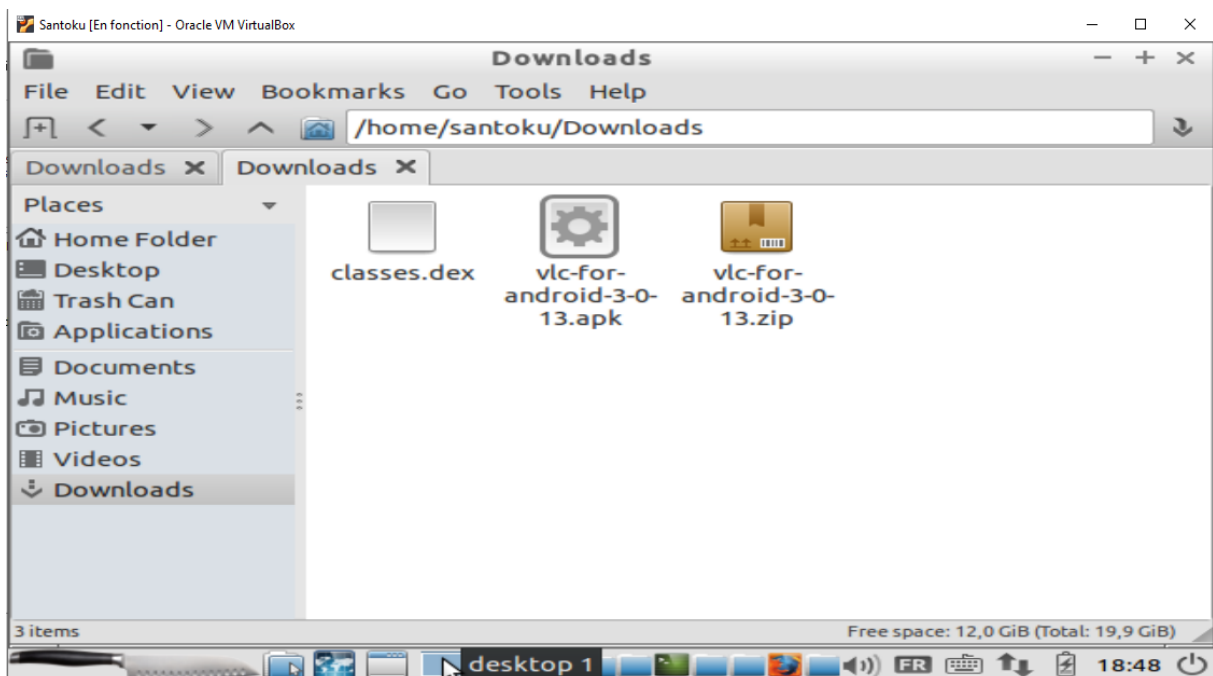
Un **décompilateur** est un outil servant à reconstituer, partiellement ou totalement, le **code source** d'un **logiciel** à partir d'un **programme exécutable** téléchargé ainsi que le montre la figure suivante : alors dans un **format binaire** ;

Le compilateur utilisé ici est le Java Decompiler déjà intégré à SANTOKU combiner au dex2jar un utilitaire dédié à la décompilation d'applications ANDROID.

Une fois le fichier téléchargé, on le retrouve dans le dossier de téléchargement par défaut de l'environnement SANTOKU tel qu'il suit :



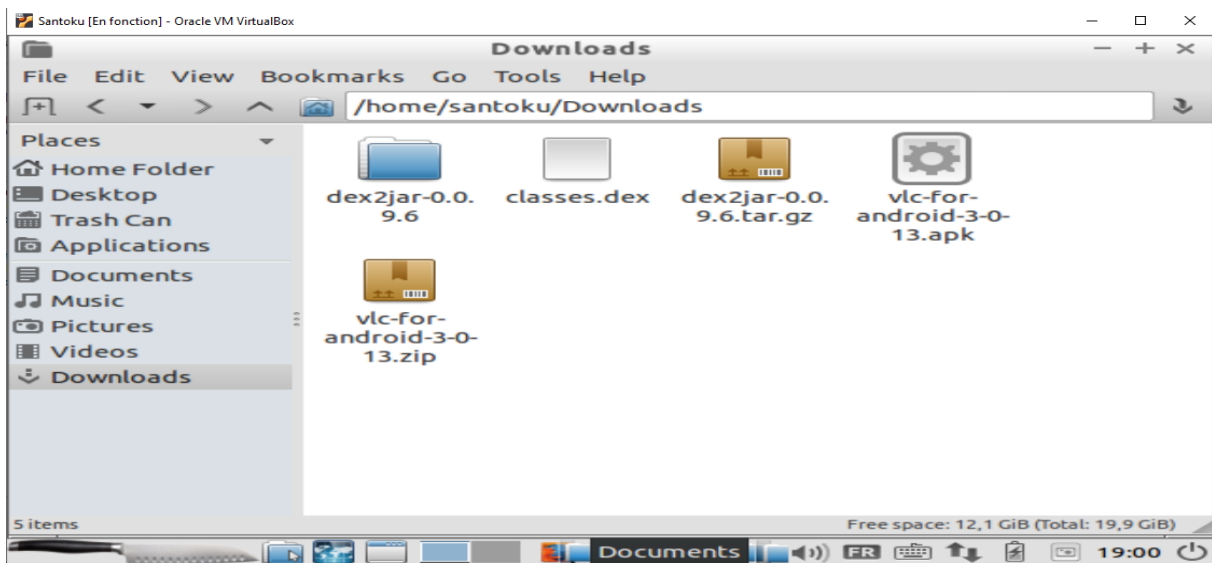
Il faut faire une copie de ce fichier au même endroit l'extension .zip, puis extraire du .zip obtenu le fichier **classes.dex**. On a la figure suivante :



Il faut ensuite télécharger le fichier dex2jar-0.0.9.6.tar.gz sur <https://sourceforge.net/projets/dex2jar/> et le décompresser dans le même dossier.



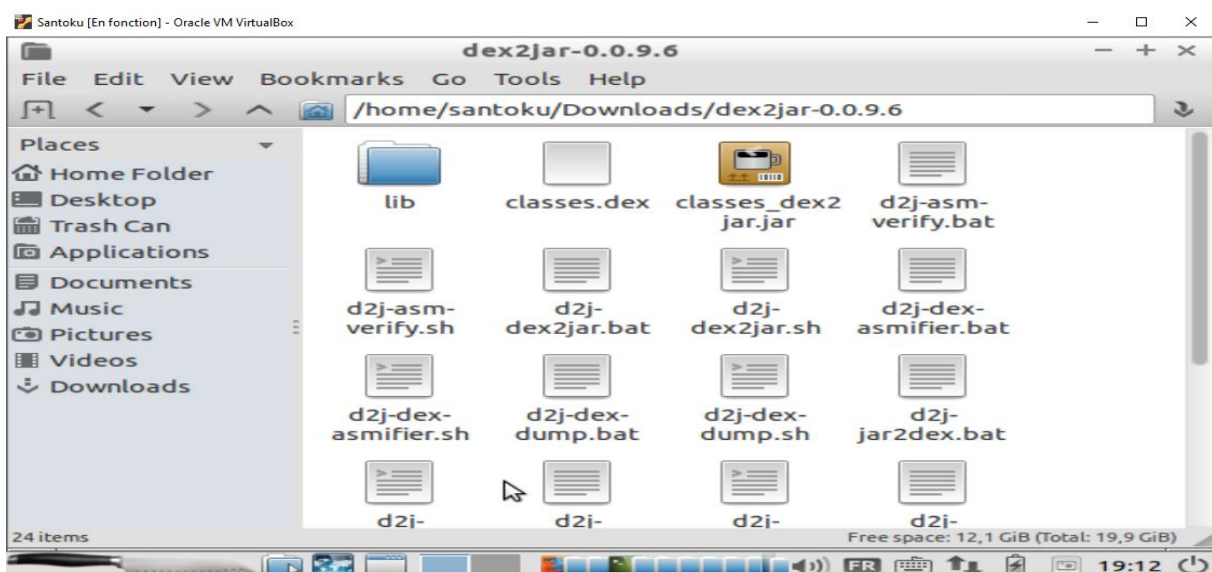
On a l'affichage suivant :



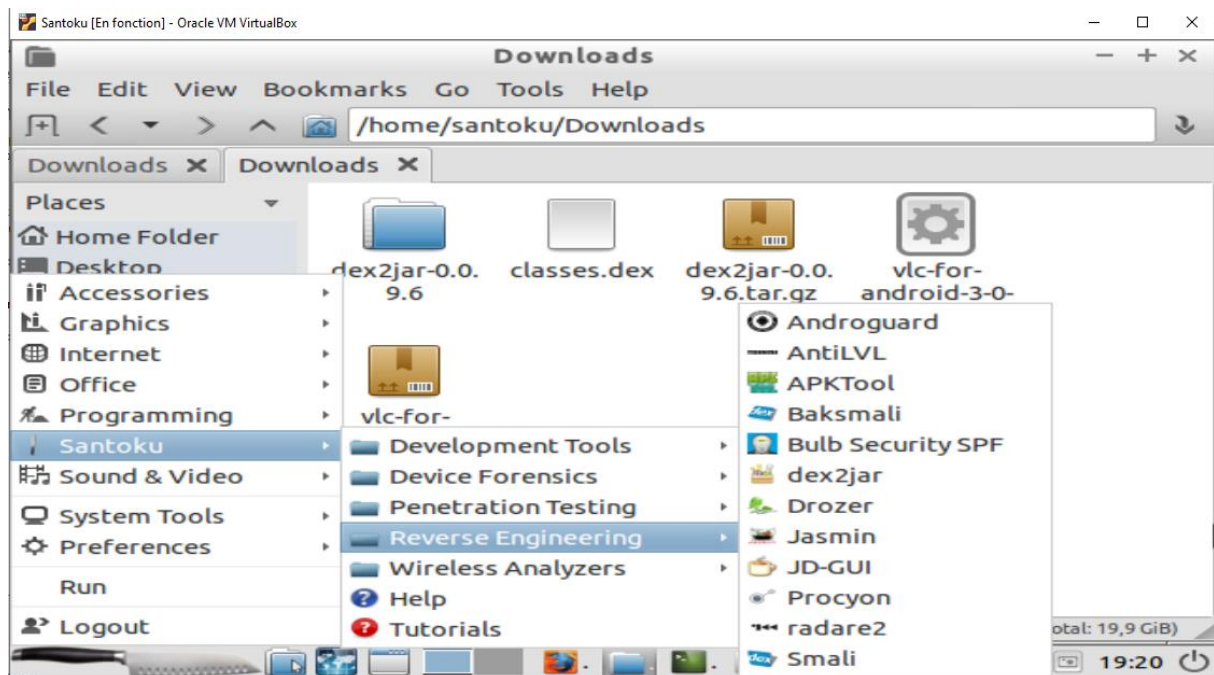
Pour la suite, il faut copier le fichier `classes.dex` dans le dossier décompressé `dex2jar-0.0.9.6` et ouvrir l'invite de commande afin d'y saisir **`cd /Downloads/dex2jar-0.0.9.6`** et **`dex2jar classes.dex`** pour obtenir la figure suivante si aucun problème rencontré.

```
santoku@santoku-VirtualBox:~/Downloads/dex2jar-0.0.9.6$ dex2jar classes.dex
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar classes.dex -> classes_dex2jar.jar
Done.
santoku@santoku-VirtualBox:~/Downloads/dex2jar-0.0.9.6$
```

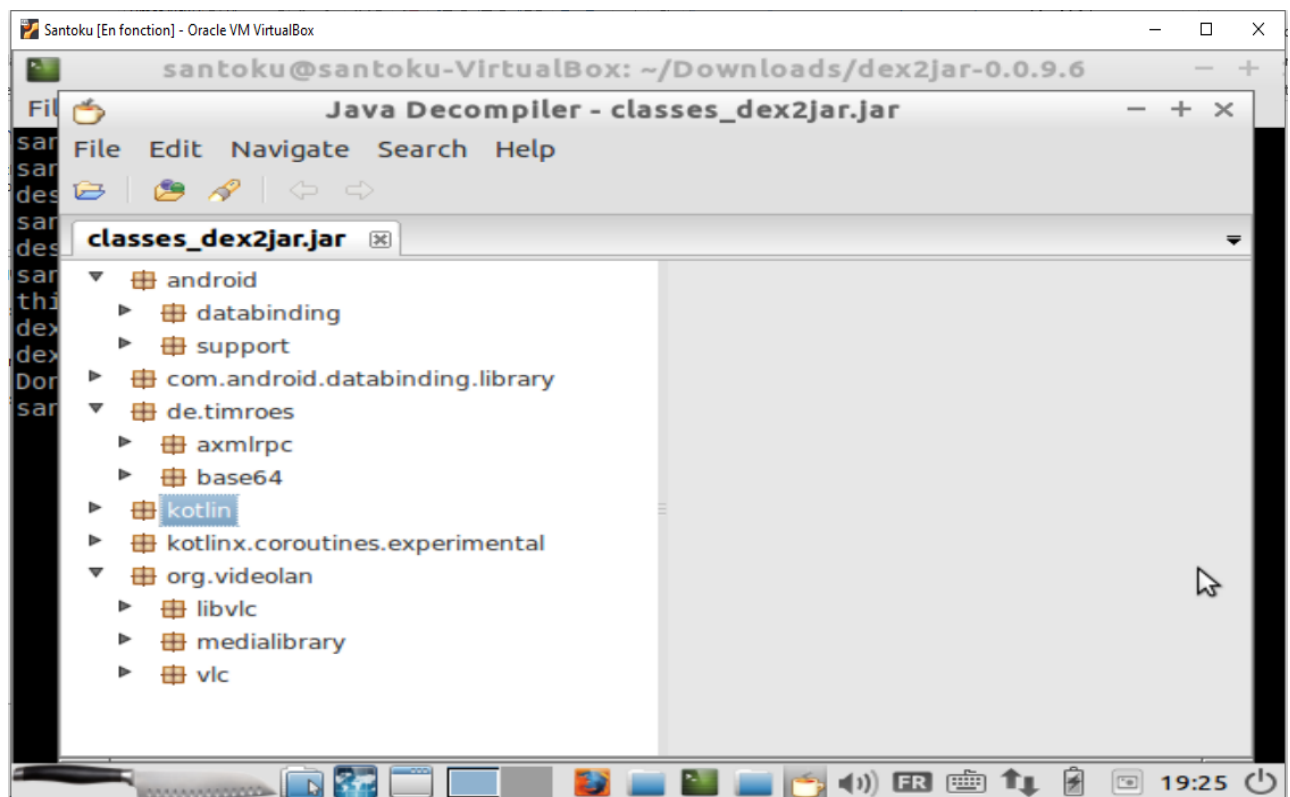
Un fichier `classes_dex2jar.jar` est généré. Il suffit d'ouvrir ce fichier avec JD-GUI de SANTOKU et voilà : on a le code source désiré.



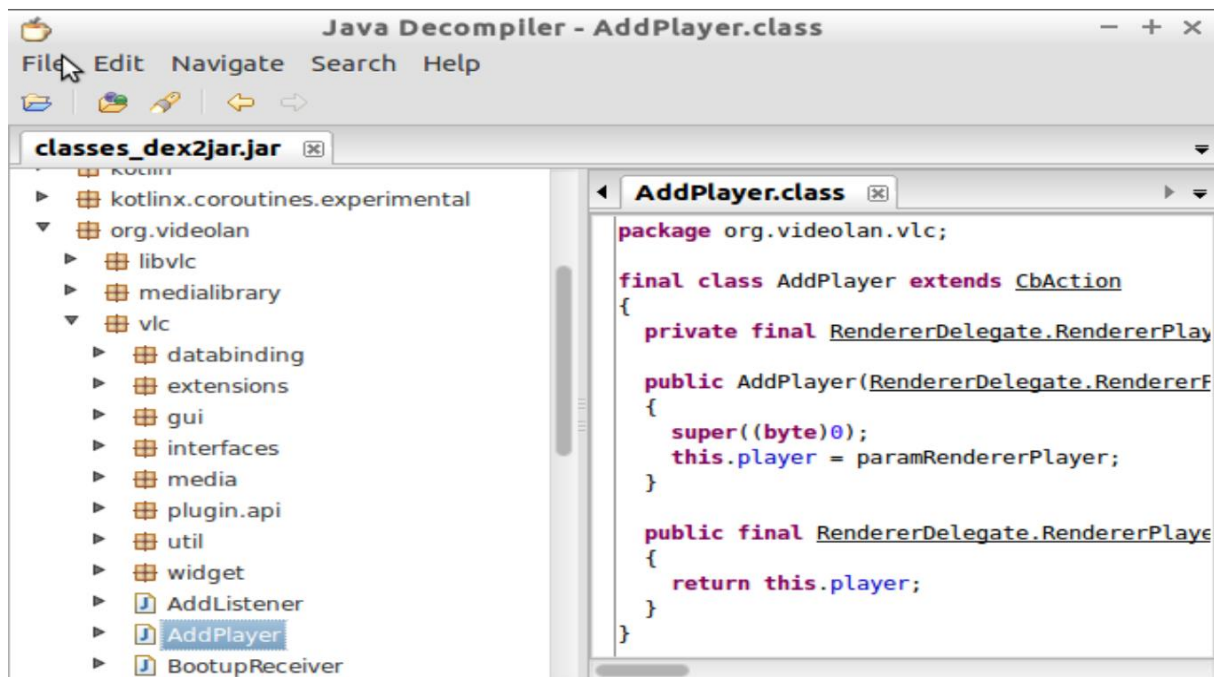
## Ouverture du JD-GUI :



Il faut faire glisser puis déposer le fichier sur la fenêtre ouverte pour avoir tous le code source de l'application :



La partie qui nous intéresse est dans le dossier vlc :

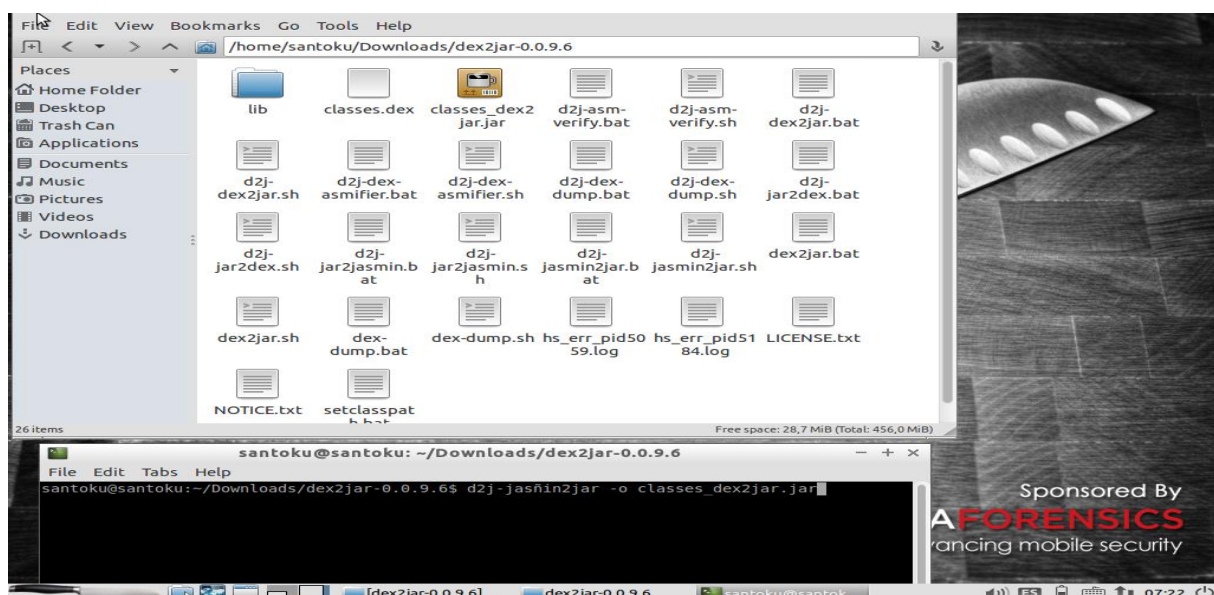


Les dossiers ici présentés possèdent des fichiers qui héritent d'une et une seule classe parmi lesquelles :

Activity, Service, BroadcastReceiver et ContentProvider qui sont les classes natives pour la production d'apk.

### 3) RECOMPILATION

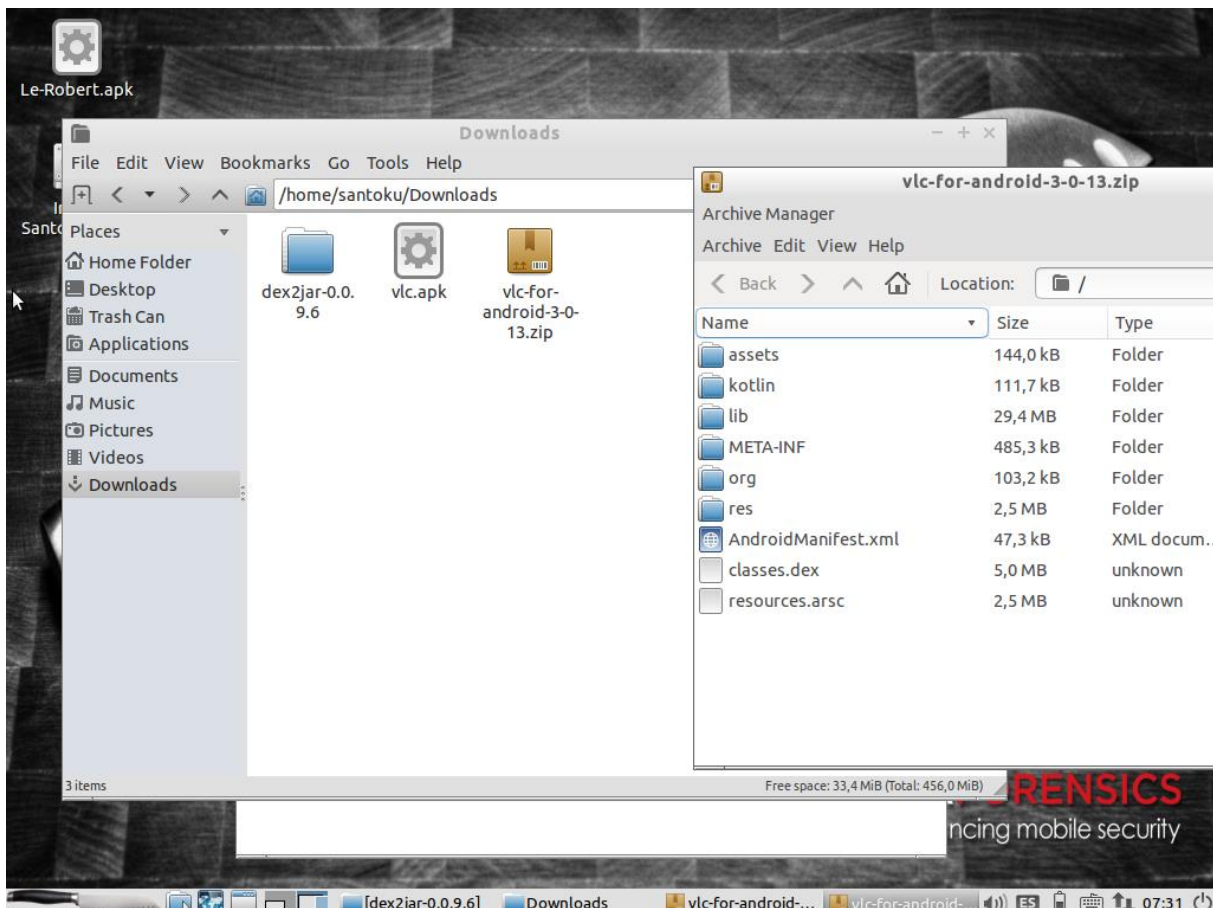
Après toute modification des fichiers .class contenus dans le 'classes\_dex2jar.jar' généré pour une personnalisation quelconque, il faut supprimer le fichier 'classes.dex' et saisir dans terminal ouvert dans le répertoire courant : « d2j-jar2dex -0 classes\_dex2jar.jar » :



Un fichier **jar2dex.dex** est générer :

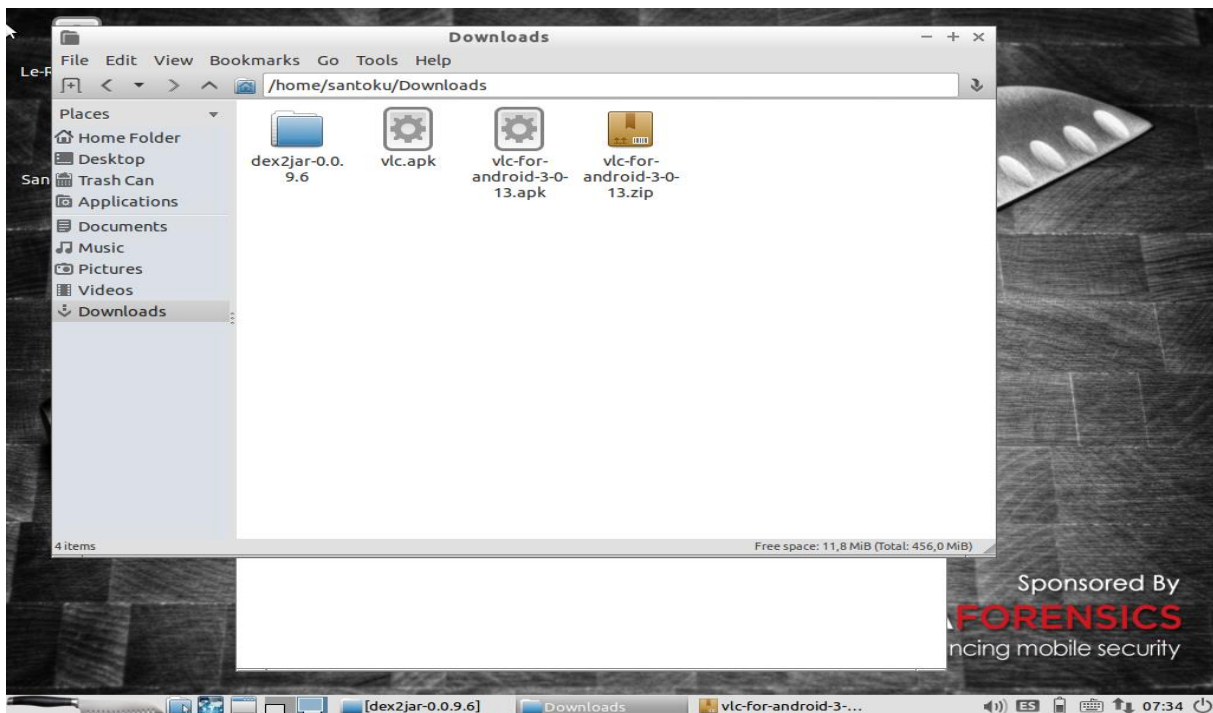


Il faut renommer le jar2dex.dex en classes.dex et remplacer celui contenu dans le dossier compressé par le nouveau (Le nôtre modifié).





Pour terminer, il faut renommer ce fichier .zip en .apk :



### Sujet 3 :

*Simuler l'inter-communication (implicite et explicite) entre deux applications Android via les intents.*

Une intention peut se traduire par « je veux, toi système, que tu fasses... ». Et cela dépend de l'action demandée et du contexte. Il faut les considérer comme de la colle entre activités permettant de les lier les unes aux autres.

Une intention est ainsi une action associée à des données. Les actions sont des constantes : ACTION\_VIEW, ACTION\_EDIT, ACTION\_PICK, les données sont des URI.

Les déclarations se déclarent dans le fichier manifest.xml de l'application.

```
< intent-filter >  
  <action android:name="android.intent.action.MAIN" />  
  <category android:name="android.intent.category.LAUNCHER" />  
< /intent-filter >
```

Android supporte les Intents explicites et implicites.

Une application peut définir les composants cibles directement dans l'Intent (Intent **explicite**) ou demander au système Android de rechercher le composant adéquat en se basant sur les données de l'Intent (Intent implicite).

**Les Intents explicites** définissent explicitement le composant qui doit être appelé par le système Android en utilisant le nom de classe Java comme identifiant.

L'exemple suivant montre comment créer un Intent explicite et comment l'envoyer au système Android. Si cette classe représente une activité, le système Android la démarrera.

```
Intent i = new Intent(this, ActivityTwo.class);  
  
i.putExtra("Value1", "This value one for ActivityTwo ");  
  
i.putExtra("Value2", "This value two ActivityTwo");
```

Les Intents explicites sont généralement utilisés au sein d'une application car les classes de cette application sont contrôlées par le développeur de l'application.

**Les Intents implicites** précisent l'action qui devrait être effectuée avec éventuellement les données pour cette action.

Par exemple, ce qui suit indique au système Android d'afficher une page Web. Tous les navigateurs Web installés doivent être enregistrés avec l'Intent correspondant en utilisant un Intent Filter.

```
Intent i = new Intent(Intent.ACTION_VIEW,Uri.parse("http://www.univ-ndere.cm")); startActivity(i);
```

Si un Intent explicite est envoyé au système Android, il recherche tous les composants qui sont enregistrés avec l'action spécifique et le type de données approprié.

Si un seul composant est trouvé, Android démarre directement ce composant. Si plusieurs composants sont identifiés par le système Android, l'utilisateur devra indiquer quel composant utiliser à l'aide d'une boîte de dialogue de sélection.

### **Transfert de données entre activités**

Un Intent contient l'action et éventuellement des données complémentaires. Le composant qui crée l'Intent peut y ajouter des données avec la méthode surchargée putExtra(). Les données supplémentaires sont des paires clé/valeur, la clé est toujours une chaîne. En valeur, vous pouvez utiliser les types de données primitifs (int, float, ..) ainsi que les types String, Bundle, Parcelable et Serializable.

Le composant récepteur peut accéder à ces informations avec les méthodes getAction() et getData() sur l'objet Intent. Cet objet Intent peut être récupéré avec la méthode getIntent().

Le composant qui reçoit l'Intent peut utiliser la méthode getIntent().getExtras () pour obtenir les données supplémentaires.

```
Bundle extras = getIntent().getExtras();

if (extras ==
null) {    return;
}

//Obtenir les données à partir de la clé String

value1 = extras.getString(Intent.EXTRA_TEXT);

if (value1 != null) {

    // faire un truc ici
}
```

**Exemple : utiliser un Intent partagé**

Beaucoup d'applications Android vous permettent de partager des données avec d'autres personnes, par exemple les applications Facebook, Google+, Gmail et Twitter. Vous pouvez aussi envoyer des données à l'un de ces composants. Le morceau de code suivant vous montre comment faire :

```
Intent intent = new Intent(Intent.ACTION_SEND);  
  
intent.setType("text/plain");  
  
intent.putExtra(android.content.Intent.EXTRA_TEXT, "News for you!");  
  
startActivity(intent);
```

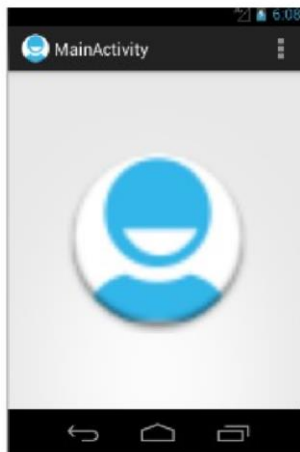
### **Récupérer le résultat d'une sous-activité**

Une activité peut être fermée avec le bouton « back » sur le téléphone. Dans ce cas, la méthode `finish()` est appelée. Si l'activité a été lancée avec la méthode `startActivity(Intent)`, l'appelant ne requiert pas de résultat ou de retour de l'activité qui est maintenant terminée.

Si vous lancez l'activité avec la méthode `startActivityForResult()`, vous attendez un retour de la sous-activité. Dès que la sous-activité se termine, la méthode `onActivityResult()` de la sous-activité est appelée et vous pouvez exécuter des actions suivant le résultat.

Dans la méthode `startActivityForResult()` vous pouvez spécifier le code de retour pour déterminer quelle activité vous avez démarrée. Ce code vous est retourné. L'activité démarrée peut aussi positionner un résultat que l'appelant peut utiliser afin de déterminer si l'activité a été abandonnée ou pas.

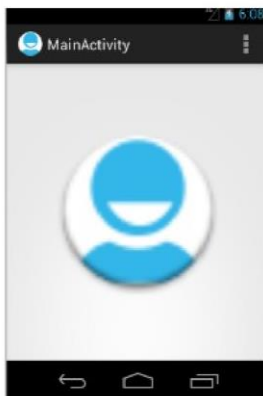




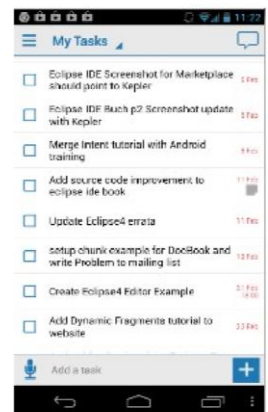
Intent resolution by the  
Android system



One activity is  
started



Intent + resultCode  
provided by called  
activity



`onActivityResult(requestCode,  
resultCode, intent)`

requestCode  
provided by Android to  
identify which activity  
type was started

La sous-activité utilise la méthode `finish()` pour créer un Intent et mettre des données dedans. Cela mettra également une donnée de retour via l'appel de la fonction `setResult()`.

### Exemple : Enregistrer une activité en tant que navigateur

Le code suivant enregistre une **activité** avec un **Intent** qui est déclenchée lorsque quelqu'un veut afficher une page Web.

```
<activity android:name=".BrowserActivitiy"
android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="http"/>
    </intent-filter>
</activity>
```

### Exemple : Enregistrer une activité pour un Intent partagé

```
< activity
    android:name=".ActivityTest"
    android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.SEND" />

        <category android:name="android.intent.category.DEFAULT" />

        <data android:mimeType="text/plain" />

    </intent-filter>
</activity>
```

Si un composant ne définit pas d'**Intent Filter**, il peut être appelé uniquement par un **Intent** explicite.

## **BIBLIOGRAPHIE :**

### **Sites Internet :**

1. <https://jaytaylor.com/notes/node/1458686144000.html>
2. <https://java.com/fr>
3. [https://fr.wikipedia.org/wiki/Application\\_\(informatique\)](https://fr.wikipedia.org/wiki/Application_(informatique))
4. <https://www.youtube.com/watch?v=5lA0MPZESss&t=52s>
5. <https://developer.android.com/docs/>
6. <https://tools.kali.org/reverse-engineering/dex2jar>

### **Documents :**

1. *Chapitre 2 du cours de Système d'exploitation Mobile 2017/2018* du Dr Franklin TCHAKOUNTE.