

LO52  
Travaux Pratiques 5:  
Introduction à l'AOSP

Professeur encadrant :  
BRISSET Fabien

Elèves :  
ROMET Pierre  
PROST Guillaume

Automne 2017

# Contents

<b>1</b>	<b>Objectif</b>	<b>2</b>
<b>2</b>	<b>Interface graphique</b>	<b>3</b>
<b>3</b>	<b>JNI</b>	<b>4</b>
3.1	JNI - Prototype de fonction . . . . .	4
3.2	JNI - Code C++ . . . . .	5

# Chapter 1

## Objectif

Dans ce Tp, il nous fut proposé de travailler avec le NDK Android. Le NDK est un ensemble d'outils, permettant d'implémenter des éléments d'un projet en utilisant un langage natif, tel que le C et/où le C++. Afin de déployer une couche JNI au sein d'une application Android. Le JNI est un mécanisme de programmation de plus en plus présent sous Android; permettant de déployer des binding Java  $\leftrightarrow$  C/C++.

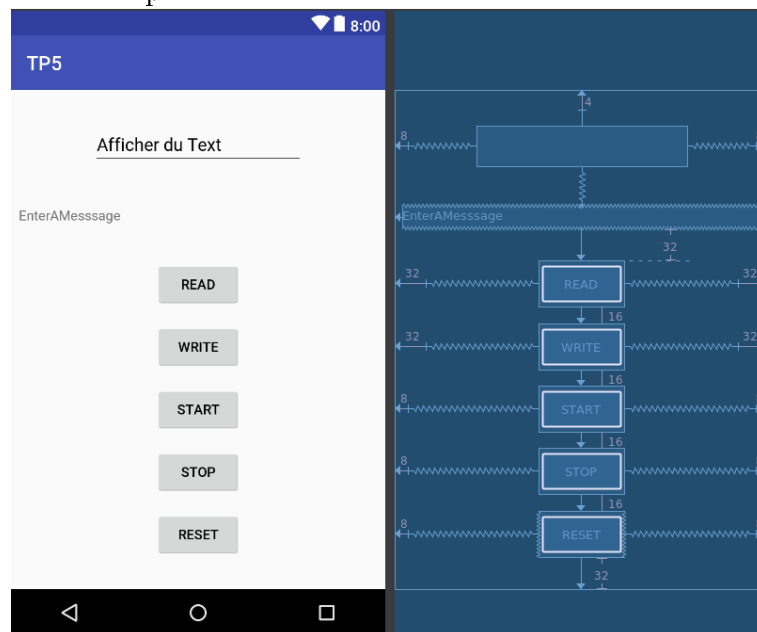
# Chapter 2

## Interface graphique

Dans un premiers temps, nous avons du déployer une application Android, avec une interface composé des éléments suivants:

- Un label
- Un champs de saisie
- boutons

Notre interface se présente donc comme suivant:



# Chapter 3

## JNI

Après avoir créé notre application android (en spécifiant lors que la création, la prise en compte de "include C++ Support"), nous nous sommes tourné vers l'intégration de la JNI au sein de notre projet.

Pour cela nous avons commencé par déclaré les prototypes des fonctions natives, (au sein d'une classe C++) que nous devons intégrer, en respectant le standard de programmation imposé par la JNI.

### 3.1 JNI - Prototype de fonction

Nous avons déployé cinq fonctions en respectant la norme suivante:

---

```
JNIEXPORT <type_retour> JNICALL
    Java_<package_tree>_<class_name>_<methode_name>
    (JNIEnv * env, jobject this, <TYPE parameter>);
```

---

Ainsi, les prototypes de nos fonctions furent déclaré comme suivant:

---

```
#include <jni.h>
/*
 * Read function
 */
JNIEXPORT jstring JNICALL
    Java_com_lo52_dewback_tp5_L052MainActivity_read
    (JNIEnv * env, jobject obj, jstring myString)

/*
 * Write function
 */
```

```

JNIEXPORT jstring JNICALL
    Java_com_lo52_dewback_tp5_L052MainActivity_write
(JNIEnv *env, jobject obj, jstring myString)

/*
 * Stop function
 */
JNIEXPORT jstring JNICALL
    Java_com_lo52_dewback_tp5_L052MainActivity_stop
(JNIEnv *env, jobject obj, jint myInt)

/*
 * Start function
 */
JNIEXPORT jstring JNICALL
    Java_com_lo52_dewback_tp5_L052MainActivity_start
(JNIEnv *env, jobject obj, jint myInt)

/*
 * Reset function
 */
JNIEXPORT jstring JNICALL
    Java_com_lo52_dewback_tp5_L052MainActivity_reset
(JNIEnv *env, jobject obj)

```

---

## 3.2 JNI - Code C++

Par la suite, au sein de notre de notre classe Java principale "LO52MainActivity.java", nous avons utilisé la fonction suivante afin de charger notre bibliothèque JNI.

```

static {
    System.loadLibrary("native-lib");
}

```

---

Puis, nous avons initialisé nos cinq fonctions comme suivant; chaque prototype étant préfixé de l'attribut "Native", qui en java spécifie que la fonction référence un code natif.

```

native String read(String pString);
native String write(String pString);
native String stop(int pInt);
native String start(int pInt);

```

```
native String reset();
```

---

Enfin, nous avons définie les fonctions, déclenché par un appui sur l'un des boutons de notre interface, nous permettant de faire appel à nos fonctions natives.

---

```
public void onRead(View pView);  
public void onWrite(View pView);  
public void onStart(View pView);  
public void onStop(View pView);  
public void onReset(View pView);
```

---