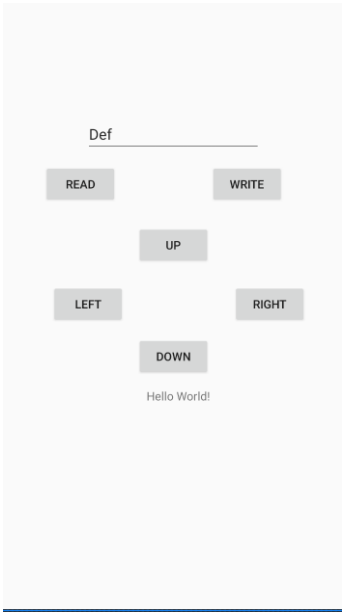
TP4 LO52

Il faut tout d'abord installer les composants nécessaires pour le NDK Android. Pour cela il faut se rendre dans Tools > SDK Manager (onglet SDK Tools) puis cocher « NDK (Side by side) » et « CMake » et enfin cliquer sur « appliquer ». S'ensuit un téléchargement et une installation des composants assez long.

J'ai ensuite créé un nouveau projet natif C++ qui contient un Hello World.

Dans l'application, j'ai commencé par placer les boutons et le textfield. Chaque bouton doit avoir un ID différent et des contraintes pour pouvoir le placer dans le layout.

Pour ce qui est des boutons de directions, j'ai créé une fonction clique() en Java qui récupère le texte du bouton cliqué puis appelle une fonction native displayDirection() qui va convertir le string Java en type char et vérifier le texte du bouton. Suivant le bouton, la fonction va retourner le texte correspondant en Japonais. A son retour, la fonction Java n'a plus qu'à afficher le texte dans le textfield. Il ne reste qu'à définir la propriété onclick() des boutons sur notre fonction clique()




```
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_tp4_MainActivity_displayDirection(
        JNIEnv* env, jobject MainActivity  /* this */, jstring buttonName) {

    const char *nativeString = env->GetStringUTFChars(buttonName,  isCopy: 0);
    if (strcmp(nativeString, "Up") == 0){return env->NewStringUTF( bytes: "アップ");}
    if (strcmp(nativeString, "Down") == 0){return env->NewStringUTF( bytes: "ダウン");}
    if (strcmp(nativeString, "Left") == 0){return env->NewStringUTF( bytes: "左");}
    if (strcmp(nativeString, "Right") == 0){return env->NewStringUTF( bytes: "正しい");}

}
```

Pour les boutons Read et Write, c'est à peu près le même principe.

- Une fonction Java readAndWrite() qui sera appelée lorsque l'on clique sur un des boutons.
- Un nombre entre 0 et 10 généré aléatoirement.
- Un appel à une fonction native différente suivant Read ou Write prenant en paramètre le nombre aléatoire.
- Deux fonctions (une pour Read, l'autre pour Write) natives qui : faire la multiplication, convertir le type int en char, concaténer le message 'READ : ' avec le nombre calculé, retourner cette valeur.
- Afficher la valeur dans le textfield.

```
extern "C" JNIEXPORT jstring JNICALL
Java_com_example_tp4_MainActivity_displayRead(
        JNIEnv* env, jobject MainActivity  /* this */, jint number) {

    jint finalNumber = number*number; //Multiplication demandée
    char buf[64]; // Création d'un char pour contenir le nombre calculé
    sprintf(buf,  format: "%d", finalNumber);  // le char contient maintenant le nombre calculé
    char msg[60] = "READ : ";
    strcat(msg, buf); // Concaténation du message avec le nombre
    return env->NewStringUTF(msg); // On envoie
}
```

Une vidéo de démonstration de l'appli est disponible ici

Le plus dur dans ce TP aura été de gérer les différences entre les types Java et C++. C'est assez sombre tout ça.