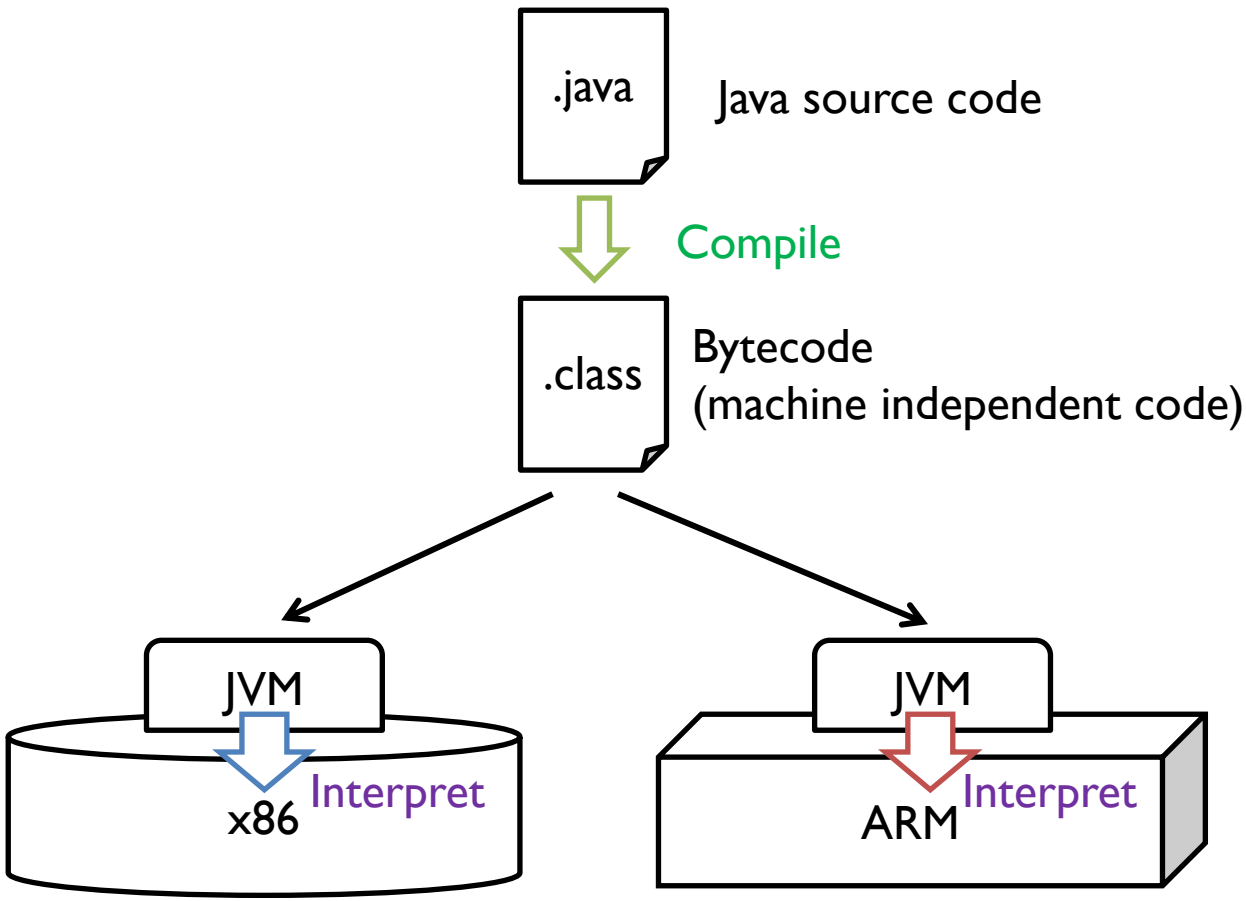
SSE3052: Embedded Systems Practice

Jinkyu jeong
jinkyu@skku.edu
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu

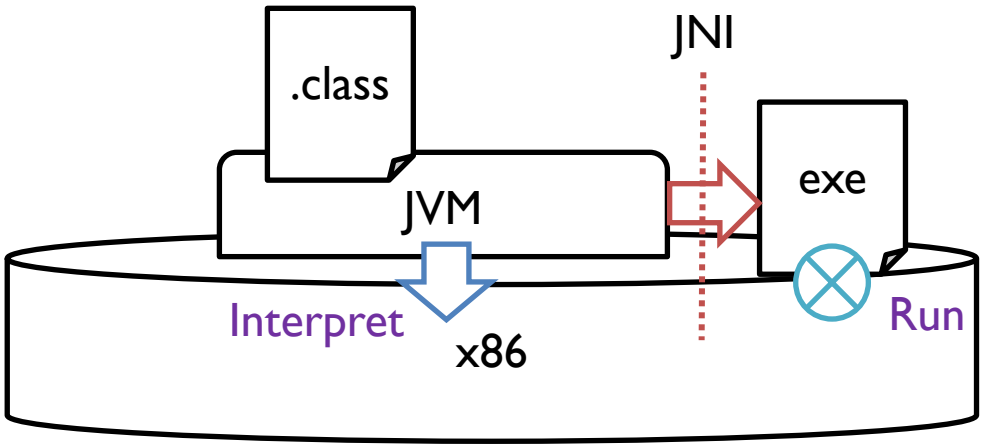
Java Runtime Environment



Java is slow!

Two techniques to speed up:

- Just-In-Time (JIT) compiler
  - Compile bytecode into native code at runtime
- **Java Native Interface (JNI)**
  - Run C/C++(native) code from Java application



Java Native Interface

- Provides interface to call C/C++ code
- JNI is useful for:
  - Run computationally intensive code (e.g. game)
  - Reuse legacy code written in C/C++
  - Implement platform-dependent features not supported in Jav a

JNI Programming

1. Write Java code
2. Compile Java code
3. Generate C/C++ header file
4. Write C/C++ code
5. Create shared library
6. Run Java program

Hello World Example

HelloWorld.java

```
public class HelloWorld
{
    public native void print();
}
```

Main.java

```
public class Main
{
    public static void main(String[] args)
    {
        System.loadLibrary("HelloWorld");
        new HelloWorld().print();
    }
}
```

Compile

```
$ javac HelloWorld.java
    ~> HelloWorld.class
$ javac Main.java
    ~> Main.class
```

Generate Header File

```
$ javah HelloWorld
    ~> HelloWorld.h
$ javac -h [dir] HelloWorld.java (if $javah command does not exists)
    ~> HelloWorld.h
```

HelloWorld.h

```
/* DO NOT EDIT THIS FILE - it is machine generated */
#include <jni.h>
/* Header for class HelloWorld */


#ifndef _Included_HelloWorld
#define _Included_HelloWorld
#ifdef __cplusplus
extern "C" {
#endif
/*
 * Class:     HelloWorld
 * Method:    print
 * Signature: ()V
 */
JNIEXPORT void JNICALL Java_HelloWorld_print
  (JNIEnv *, jobject);


#ifdef __cplusplus
}
#endif
#endif
```

HelloWorld.c

```
#include <stdio.H>
#include "HelloWorld.h"

JNIEXPORT void JNICALL Java_HelloWorld_print(JNIEnv *env, jobject obj)
{
    printf("Hello World\n");
    return;
}
```

Function declaration

JNIEXPORT *returnType* **JNICALL** Java_*className*_*methodName*
(JNIEnv *, jobject, …)

Create Shared Library

```
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

gcc -shared -fPIC -I $JAVA_HOME/include -I $JAVA_HOME/include/linux -o libHelloWorld.so HelloWorld.c
```

Run

```
$ export LD_LIBRARY_PATH=.

$ java Main
```

Primitive Data Types

Java Type	Native Type	Description
`boolean`	`jboolean`	Unsigned 8 bits
`byte`	`jbyte`	Signed 8 bits
`char`	`jchar`	Unsigned 16 bits
`short`	`jshort`	Signed 16 bits
`int`	`jint`	Signed 32 bits
`long`	`jlong`	Signed 32 bits
`float`	`jfloat`	32 bits
`double`	`jdouble`	64 bits
`void`	`void`	N/A

Reference Types

- `jobject`
  - `jclass`
  - `jstring`
  - `jarray`
    - `jobjectArray`
    - `jbooleanArray`
    - `jbyteArray`
    - `jcharArray`
    - `jshortArray`
    - `jintArray`
    - `jlongArray`
    - `jfloatArray`
    - `jdoubleArray`

- `public native String stringMethod(String text);`
- `JNIEXPORT jstring JNICALL Java_className_stringMethod(JNIEnv *env, jobject jstring);`

- `public native int intArrayMethod(int[] intArray);`
- `JNIEXPORT jint JNICALL Java_className_intArrayMethod(JNIEnv *env, jobject jintArray);`

- `public static native void staticMethod();`
- `JNIEXPORT void JNICALL Java_className_staticMethod(JNIEnv *env, jobject jclass);`

Access String

```
JNIEXPORT jstring JNICALL Java_className_stringMethod
  (JNIEnv *env, jobject obj, jstring string) {
    const char *str = (*env)->GetStringUTFChars(env, string, 0);
    char cap[128];
    strcpy(cap, str);
    (*env)->ReleaseStringUTFChars(env, string, str);
    return (*env)->NewStringUTF(env, strupr(cap));
}
```

Access Array

```
JNIEXPORT jint JNICALL Java_className_intArrayMethod
  (JNIEnv *env, jobject obj, jintArray array) {
    int i, sum = 0;
    jsize len = (*env)->GetArrayLength(env, array);
    jint *body = (*env)->GetIntArrayElements(env, array, 0);
    for (i=0; i<len; i++)
    {   sum += body[i];
    }
    (*env)->ReleaseIntArrayElements(env, array, body, 0);
    return sum;
}
```

Exercise 1

Implement the following methods in C code

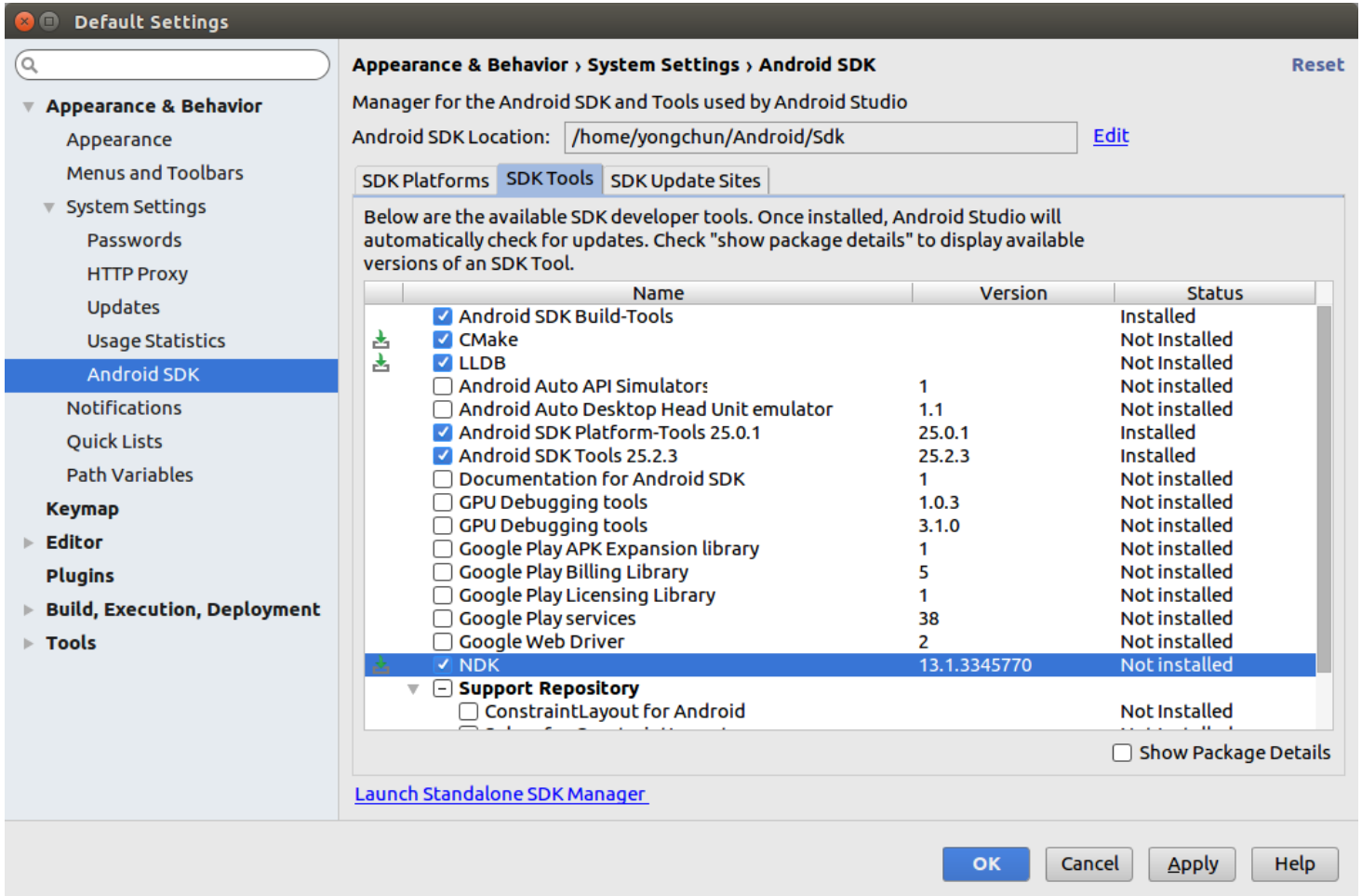
- `public native int sum(int x, int y);`
- `public native int difference(int x, int y);`
- `public native int product(int x, int y);`
- `public native double average(int x, int y);`

Android w/ JNI

- Native Development Kit (NDK) is a set of tools that allows you to use C/C++ code with Android

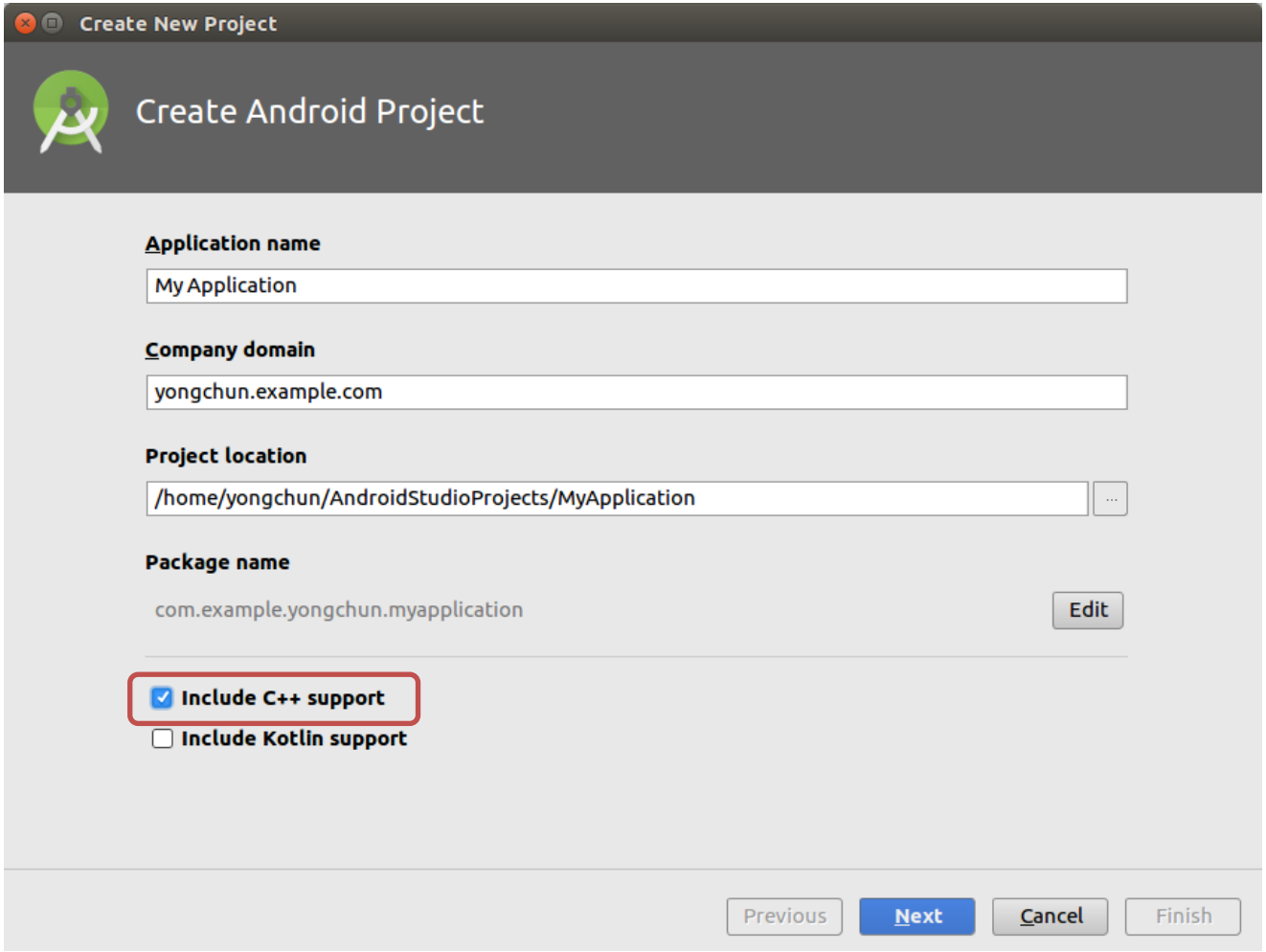
Download NDK (Android Studio)

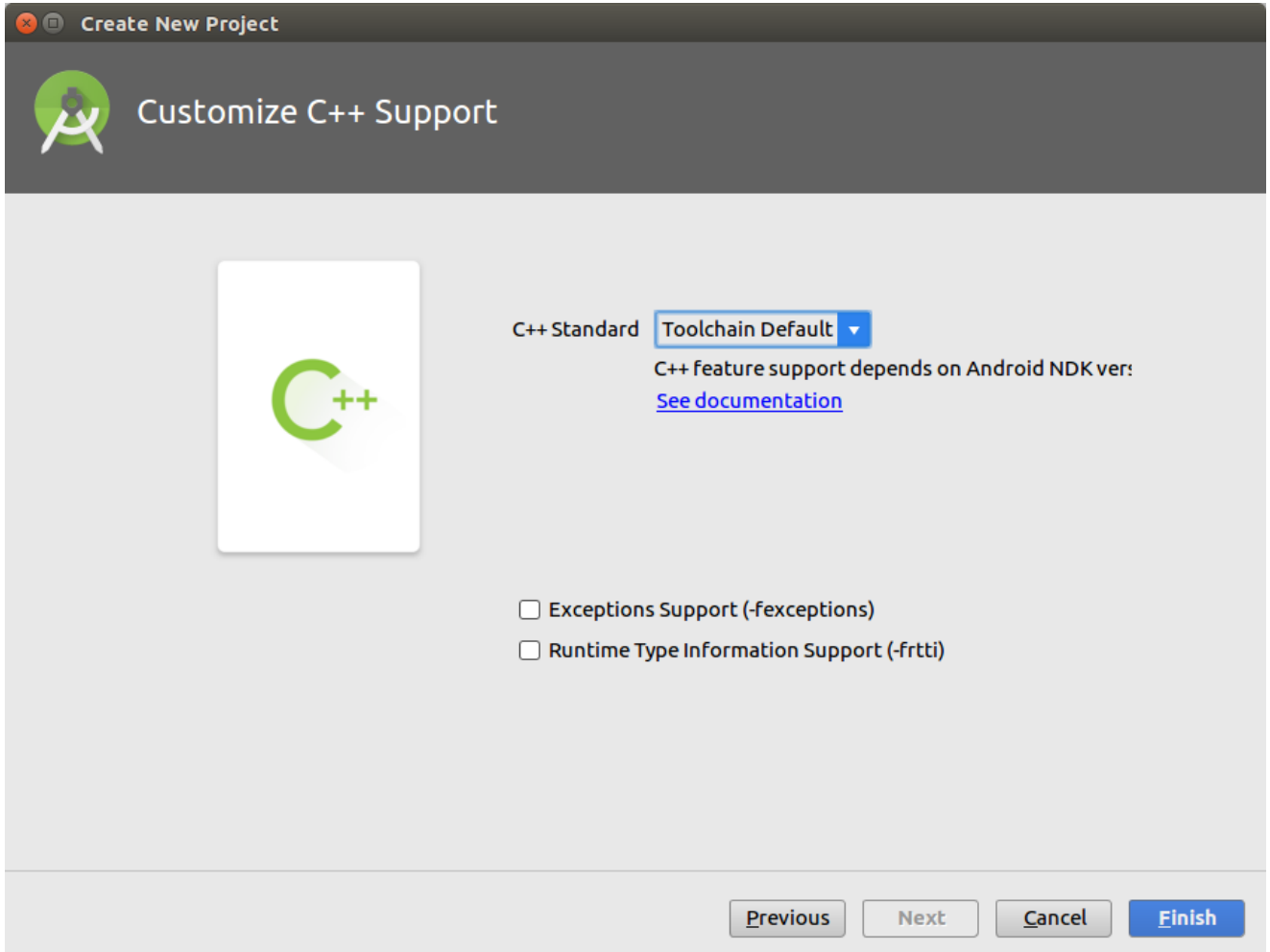
- Select `Tools` -> `SDK Manager`
- Click `SDK Tools` tab
- Check `CMake`, `NDK`



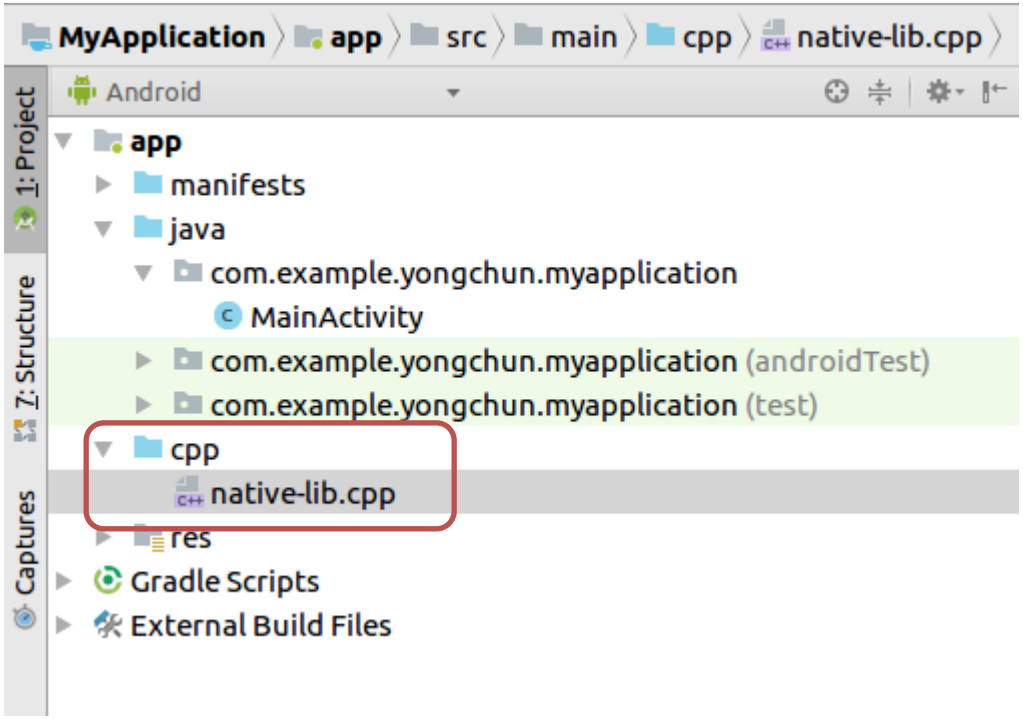
- You don’t need to install LLDB

Create New Project with C/C++




Create New Project
Customize C++ Support
C++ Standard
Toolchain Default
C++ feature support depends on Android NDK vers
See documentation
Exceptions Support (-fexceptions)
Runtime Type Information Support (-frtti)
Previous
Next
Cancel
Finish

Project Structure



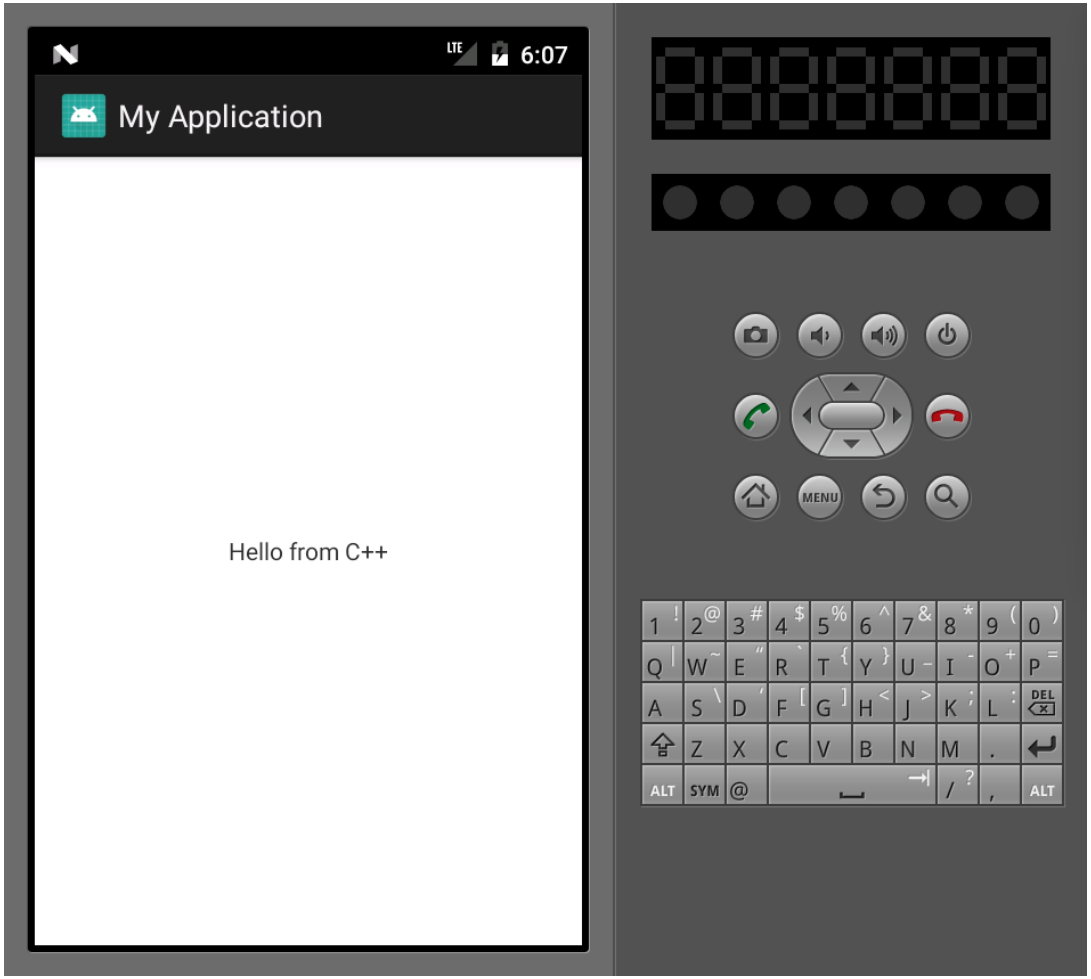
MainActivity.java

```java
public class MainActivity extends Activity {


    // Used to load the 'native-lib' library on application startup.
    static {
        System.loadLibrary("native-lib");
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        // Example of a call to a native method
        TextView tv = (TextView) findViewById(R.id.sample_text);
        tv.setText(stringFromJNI());
    }


    /**
     * A native method that is implemented by the 'native-lib' native library,
     * which is packaged with this application.
     */
    public native String stringFromJNI();
}
```

native-lib.cpp

```
#include <jni.h>
#include <string>

extern "C" JNIEXPORT jstring JNICALL
Java_com_example_yongchun_myapplication_MainActivity_stringFromJNI(
        JNIEnv *env,
        jobject /* this */) {
    std::string hello = "Hello from C++";
    return env->NewStringUTF(hello.c_str());
}
```

Run

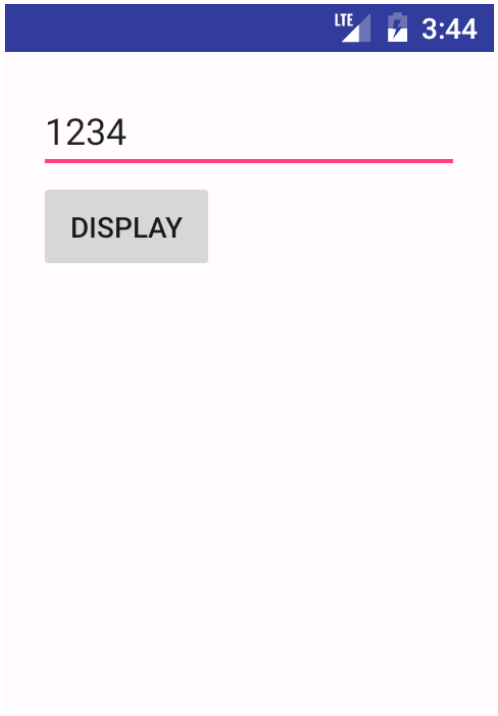


Reference

- http://www.ibm.com/developerworks/java/tutorials/j-jni/j-jni.html
- https://developer.android.com/ndk/guides/index.html
- https://developer.android.com/studio/projects/add-native-code.html

Exercise 2

- Write an Android application that takes an integer as a input a nd display the integer to 7 segment display.



Submission

- Format: YourStudentID_lab11.pdf
- Upload it on iCampus
- Due: 5/10 (Sun.) 23:59