Session: 18

Android Native Development Kit (NDK)

Objectives

- ❑ Describe app components
- ❑ Explain the process to install the NDK
- ❑ Explain the steps to create and test a sample native app

Introduction to NDK

Native Development Kit

- A set of tools
- Allows developers to use or embed native C and C++ code in Android apps
- Allows apps to be ported across platforms
- Allows reuse of existing libraries in apps

App Components 1-4

- ❑ Components that can be used when building native applications for Android devices.

Application Binary Interface (ABI)

- CPU-specific interface between system and app code.
- Facilitates interaction of app code with the system at runtime.
- App to work on systems with different CPUs and instruction sets.

Java

- During build, all Java files in the android app are converted into .dex files.
- If application does not include any Java code, a .dex file is generated for the native component.

Java Native Interface (JNI)

Programming framework that enables Java code and C or C++ code contained in an application to interact with each other.

App Components 2-4

Manifest

- NativeActivity class must be declared in the manifest if there is no Java code in the app.

ndk-build

- Shell script which runs the required NDK build scripts
- Generates binary files to be copied to the app's project path
- Requires Application.mk to build apps using the ndk-build file
- Android.mk configuration file added in the jni folder

Native Shared Libraries

- Built from the app’s native source code and have an extension of .so

Native Static Libraries

- Can be linked with other existing libraries.
- Have an extension of .a

App Components 3-4

The steps for developing an Android app:

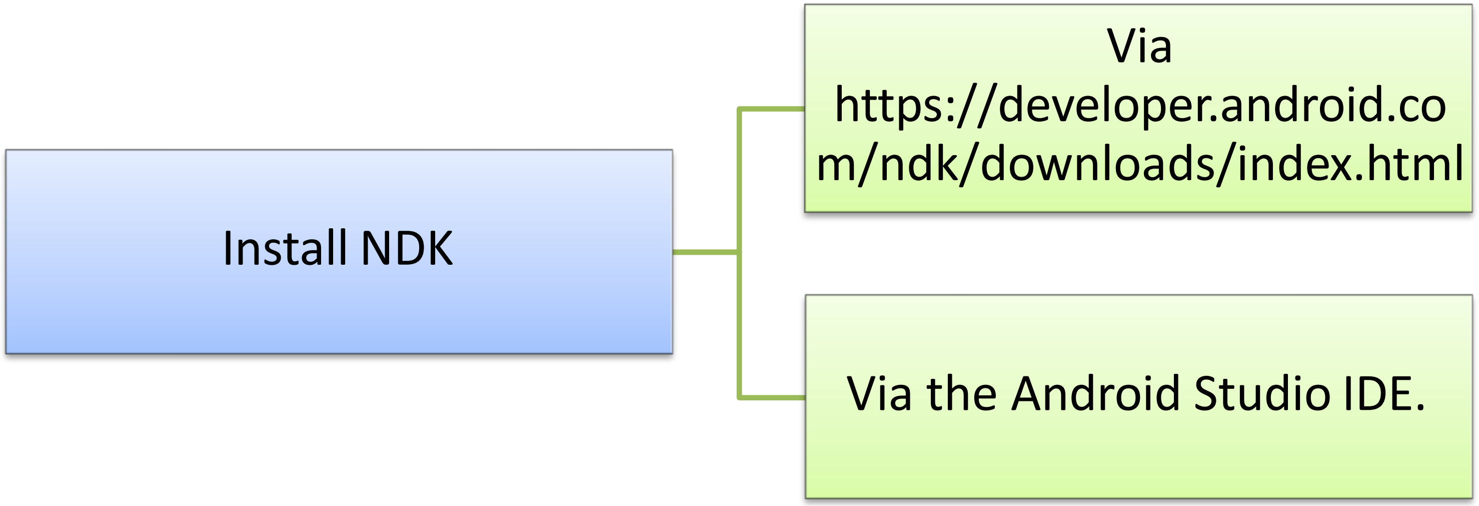
1. Design the app.
2. Decide the Java and native code components that need to be used in the app.
3. Create an Android app project.
4. For a native-only app, declare the **NativeActivity** class in **AndroidManifest.xml**.
5. Create an **Android.mk** file.
6. Add the native source code under the project's **jni** directory.
7. Use **ndk-build** to compile the native libraries.
8. Build the Java component.
9. Package all the components into an **Android Application Package (APK)** file.

App Components 4-4

Native Activity

- The NativeActivity helper class:
  - Allows to build a native activity.
  - Handles the communication between the Android framework and the app's native code.
- The application must be declared as 'native' in the AndroidManifest.xml file.

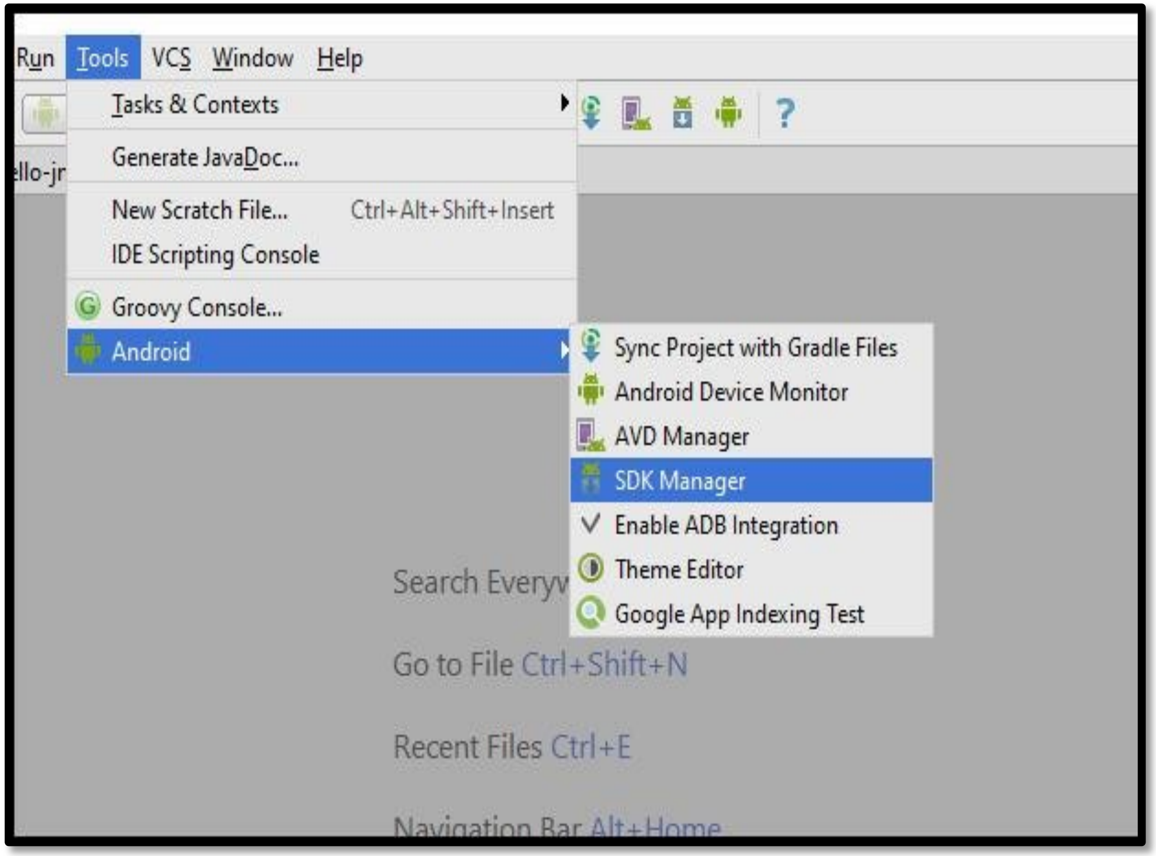
Installing the NDK 1-3


Install NDK
Via https://developer.android.com/ndk/downloads/index.html
Via the Android Studio IDE.

Installing the NDK 2-3

Alternate approach to install NDK

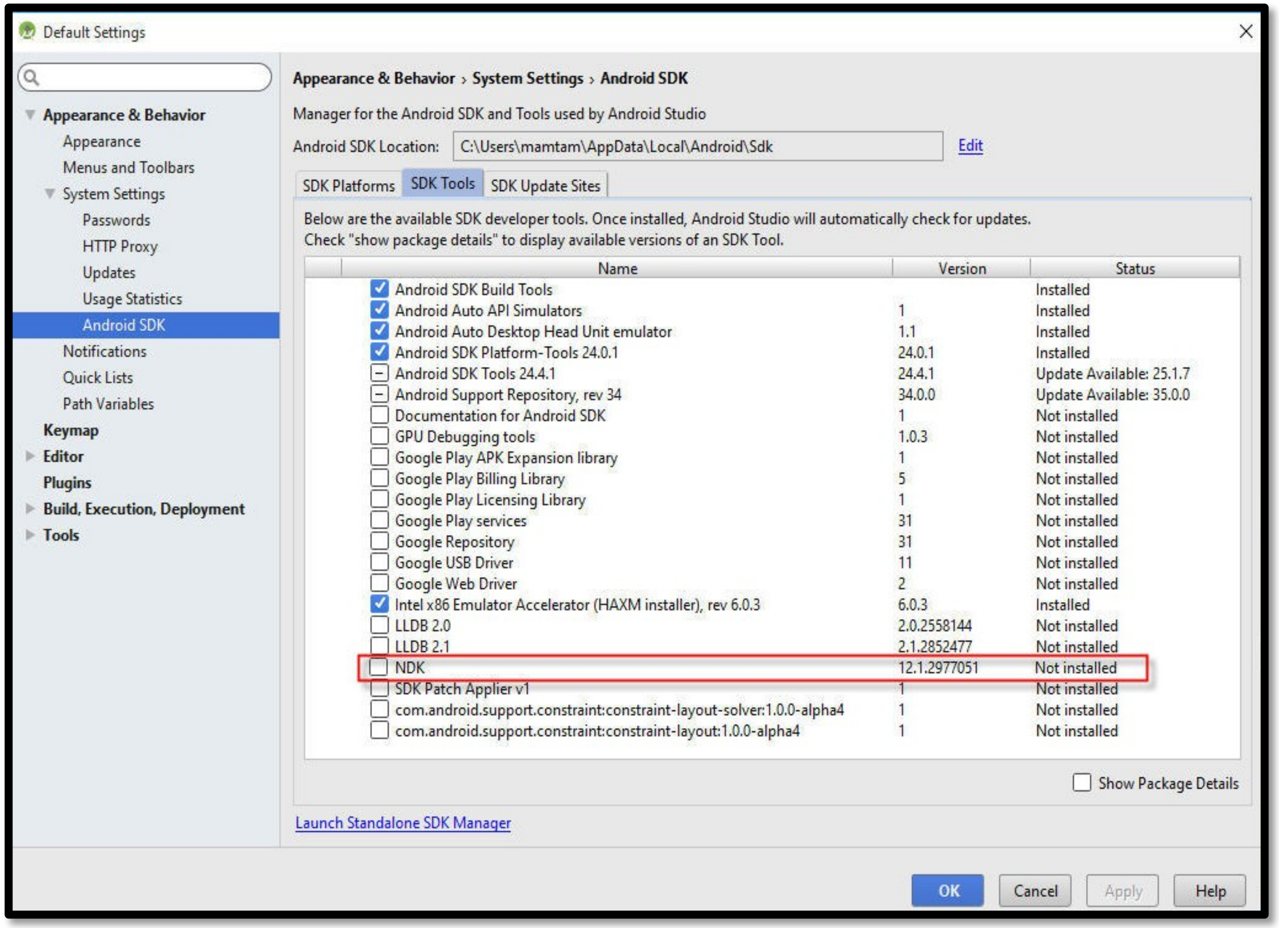
Following figure shows how to launch Android Studio:



Installing the NDK 3-3

Alternate approach to install NDK

Following figure shows the location of NDK in the SDK tools tab.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\mamtam\AppData\Local\Android\Sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

	Name	Version	Status
✓	Android SDK Build Tools		Installed
✓	Android Auto API Simulators	1	Installed
✓	Android Auto Desktop Head Unit emulator	1.1	Installed
✓	Android SDK Platform-Tools 24.0.1	24.0.1	Installed
−	Android SDK Tools 24.4.1	24.4.1	Update Available: 25.1.7
−	Android Support Repository, rev 34	34.0.0	Update Available: 35.0.0
☐	Documentation for Android SDK	1	Not installed
☐	GPU Debugging tools	1.0.3	Not installed
☐	Google Play APK Expansion library	1	Not installed
☐	Google Play Billing Library	5	Not installed
☐	Google Play Licensing Library	1	Not installed
☐	Google Play services	31	Not installed
☐	Google Repository	31	Not installed
☐	Google USB Driver	11	Not installed
☐	Google Web Driver	2	Not installed
✓	Intel x86 Emulator Accelerator (HAXM installer), rev 6.0.3	6.0.3	Installed
☐	LLDB 2.0	2.0.2558144	Not installed
☐	LLDB 2.1	2.1.2852477	Not installed
☐	NDK	12.1.2977051	Not installed
☐	SDK Patch Applier v1	1	Not installed
☐	com.android.support.constraint:constraint-layout-solver:1.0.0-alpha4	1	Not installed
☐	com.android.support.constraint:constraint-layout:1.0.0-alpha4	1	Not installed

Creating and Testing Sample App 1-8

There are two ways for developers to create applications using NDK.

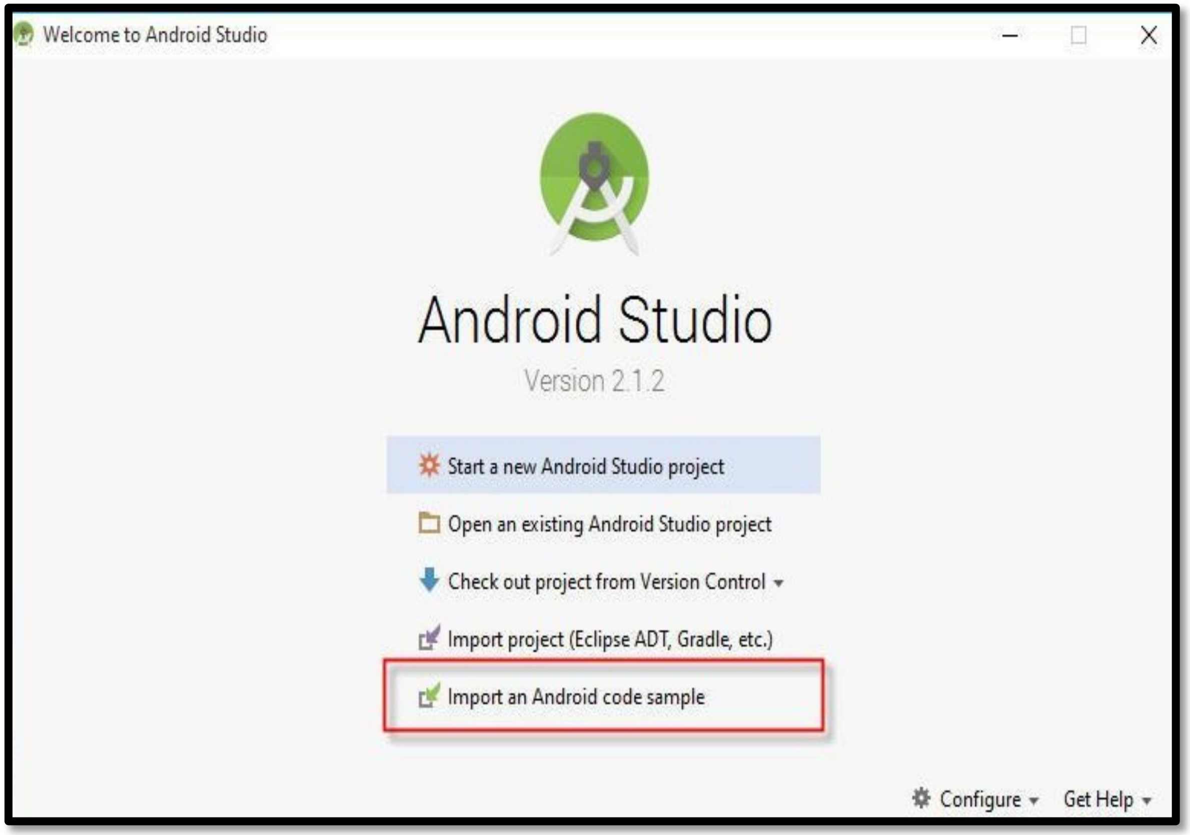
- ❑ The developers can build the application in Java or XML using the Android NDK.
- ❑ Then they can use JNI to access the Application Programming Interfaces (APIs) implemented in C or C++ using the Android NDK.

OR

- ❑ The developers can develop a native activity in C or C++.

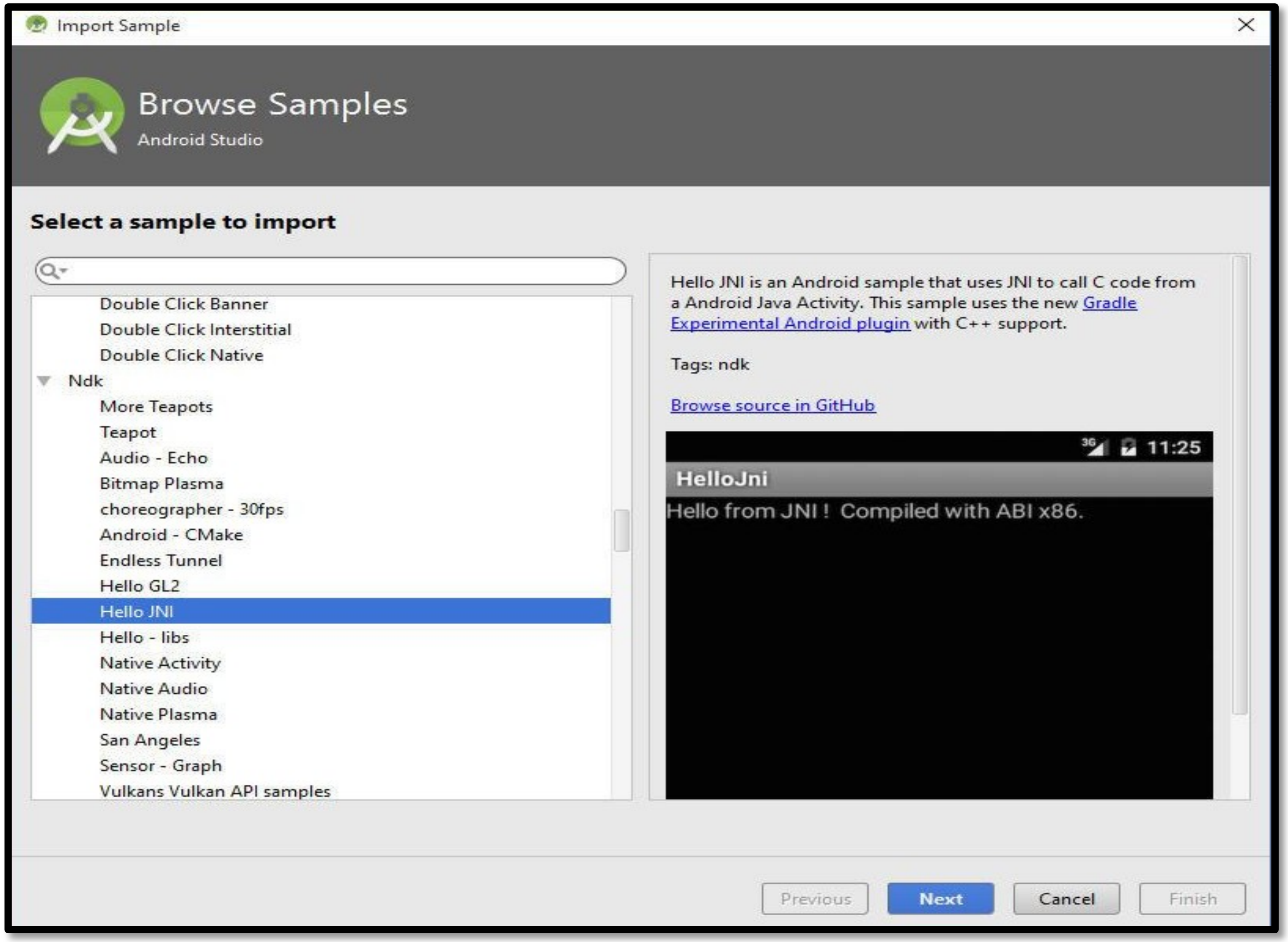
Creating and Testing Sample App 2-8

- Following figure launches Android Studio and selects the Import an Android code sample option:


Welcome to Android Studio
Android Studio
Version 2.1.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

Creating and Testing Sample App 3-8

❑ Following figure shows how to select Hello JNI:


Import Sample

Browse Samples
Android Studio

Select a sample to import

Double Click Banner
Double Click Interstitial
Double Click Native
Ndk
More Teapots
Teapot
Audio - Echo
Bitmap Plasma
choreographer - 30fps
Android - CMake
Endless Tunnel
Hello GL2
Hello JNI
Hello - libs
Native Activity
Native Audio
Native Plasma
San Angeles
Sensor - Graph
Vulkans Vulkan API samples

Hello JNI is an Android sample that uses JNI to call C code from a Android Java Activity. This sample uses the new Gradle Experimental Android plugin with C++ support.

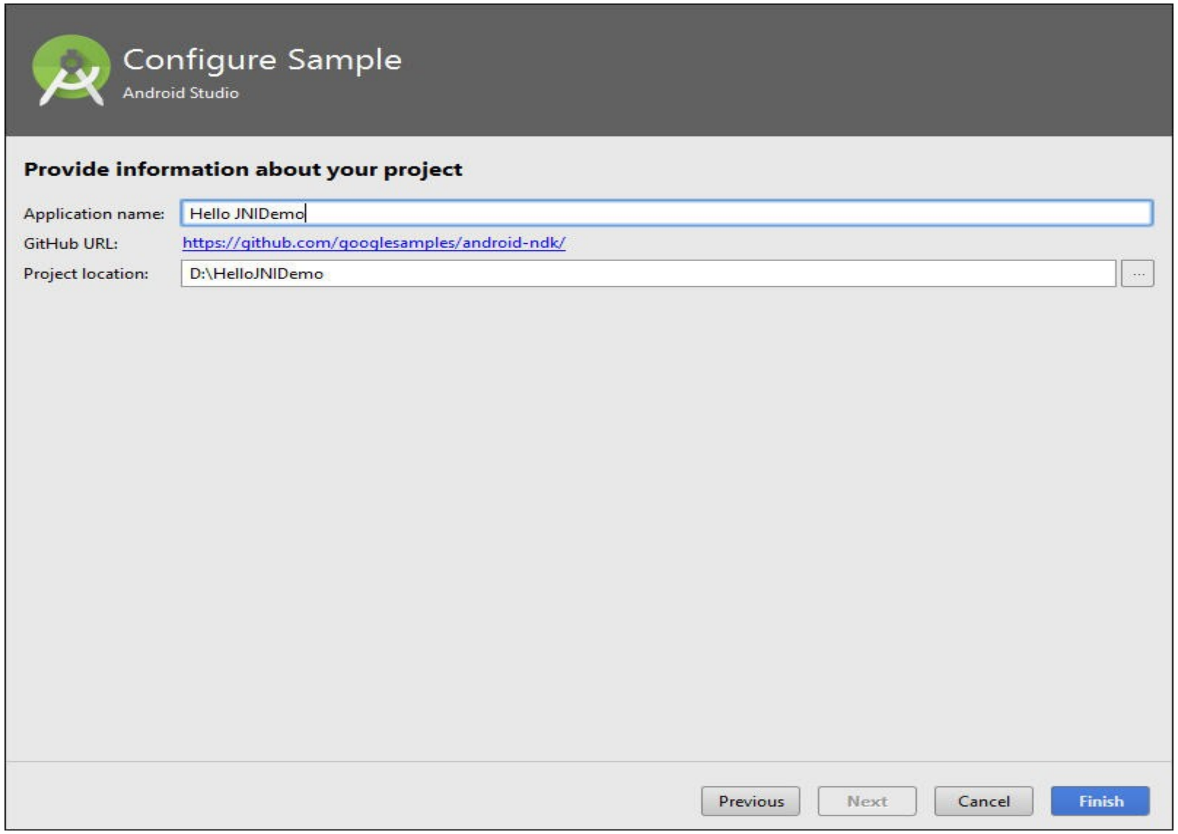
Tags: ndk

Browse source in GitHub

HelloJni
Hello from JNI ! Compiled with ABI x86.

Creating and Testing Sample App 4-8

❑ Following figure shows how to specify application name:

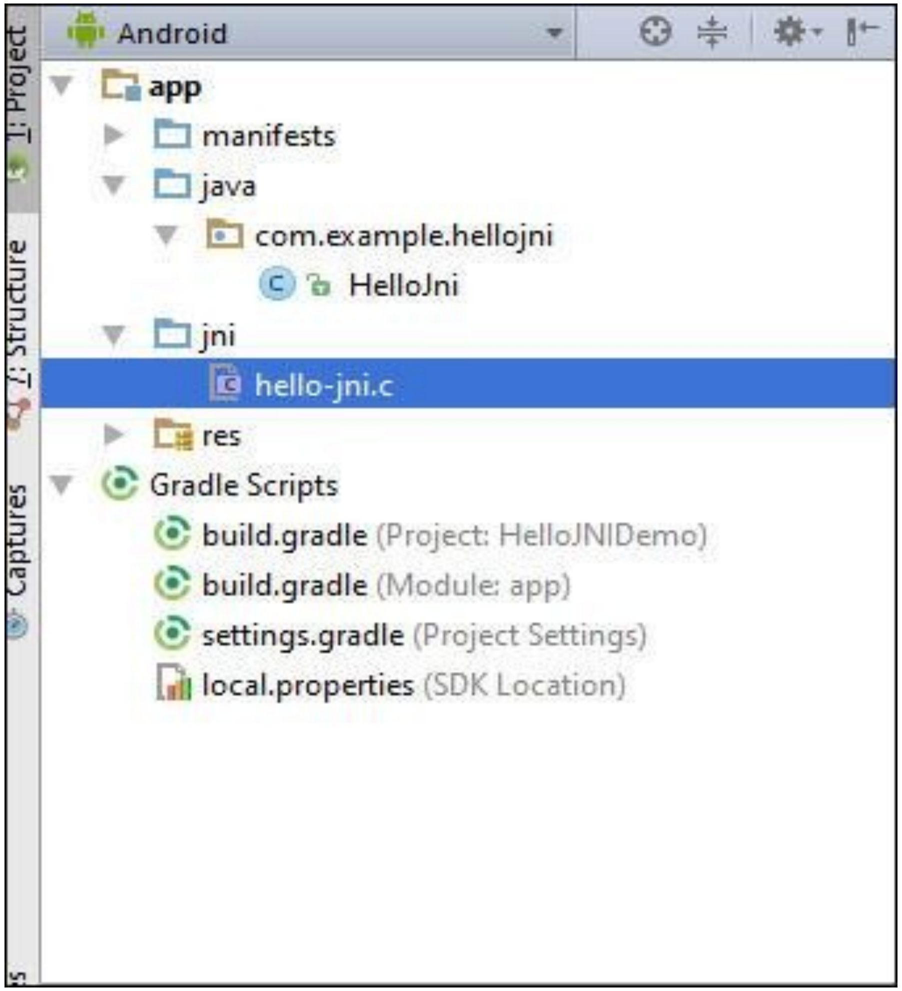

Configure Sample
Android Studio

Provide information about your project

Application name: Hello JNIDemo
GitHub URL: https://github.com/googlesamples/android-ndk/
Project location: D:\HelloJNIDemo

Creating and Testing Sample App 5-8

❑ Following figure displays the imported files:



```
app
  manifests
  java
    com.example.hellojni
      HelloJni
  jni
    hello-jni.c
  res
Gradle Scripts
  build.gradle (Project: HelloJNIDemo)
  build.gradle (Module: app)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

Creating and Testing Sample App 6-8

❑ Following code snippet shows the code in HelloJni.java:

```
public class HelloJni extends Activity
{
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    /* Create a TextView and set its content. the text is retrieved by
    •  calling a native function.*/

    TextView tv = new TextView(this);
    tv.setText( stringFromJNI() );
    setContentView(tv);
}
/* A native method that is implemented by the ‘hello-jni’ native library,
* which is packaged with this application.
*/
```

Creating and Testing Sample App 7-8

- Following code snippet shows the code in HelloJni.java:

```
public native String stringFromJNI();
    /* This is another native method declaration that is *not*
    * implemented by 'hello-jni'. This is simply to show that
    * you can declare as many native methods in your Java code
    * as you want, their implementation is searched in the
    * currently loaded native libraries only the first time
    * you call them.
    *
    * Trying to call this function will result in a
    * java.lang.UnsatisfiedLinkError exception !
    */
public native String unimplementedStringFromJNI();
    /* this is used to load the 'hello-jni' library on application
    * startup. The library has already been unpacked into
    * /data/data/com.example.hellojni/lib/libhello-jni.so at
    * installation time by the package manager.
    */
    static
    {
        System.loadLibrary("hello-jni");
    }
}
```

Creating and Testing Sample App 8-8

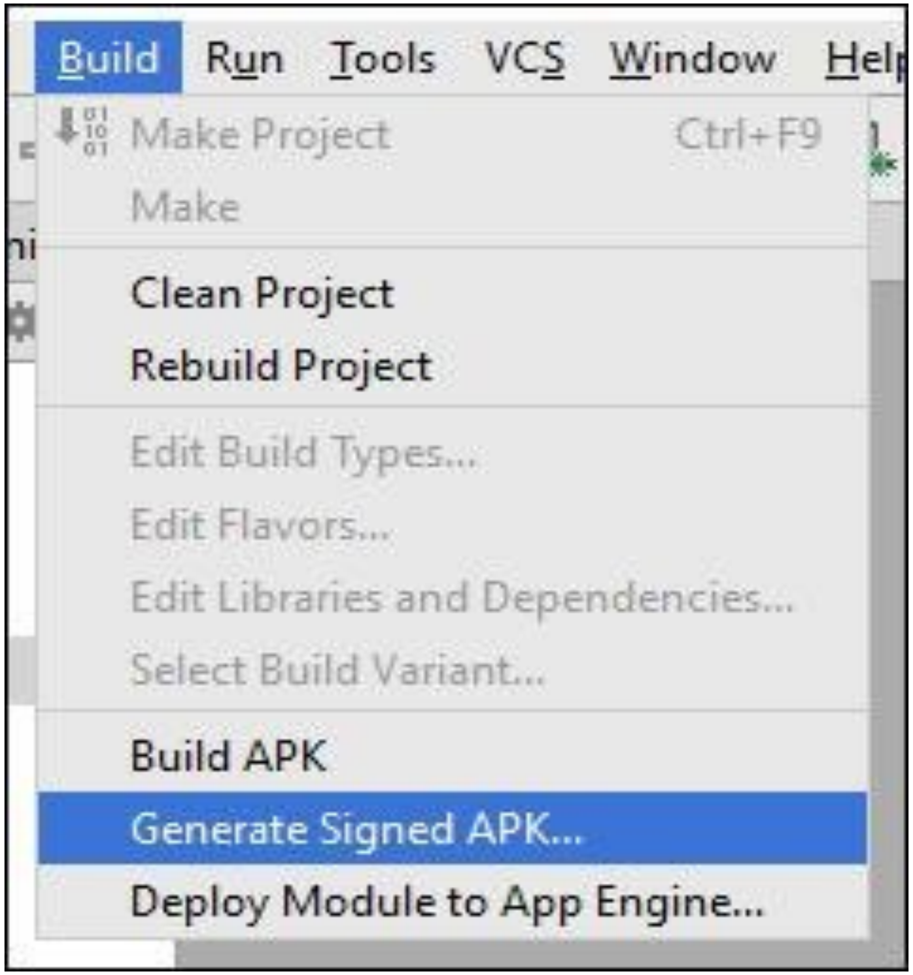
Java Native Interface

Enables Java applications to interact with native code.

Enables the C/C++ code to call Java code.

Building an .apk File

❑ Following figure shows options to build an .apk file:



Summary

- ❑ The ndk-build file automatically identifies the project that needs to be built.
- ❑ The NDK builds the native shared libraries from the app's native source code.
- ❑ The NDK builds static libraries that can be linked with other existing libraries.
- ❑ .JNI is the programming framework that enables Java code and C or C++ applications to interact with each other.
- ❑ Android Studio provides support to install NDK.
- ❑ ABI is the interface between two programs that are at different levels.
- ❑ The Android NDK has a helper class called the NativeActivity class that allows the developer to build a native activity.