

Android Application Development



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

For Aptech Center Use Only

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

For Aptech Center Use Only

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:

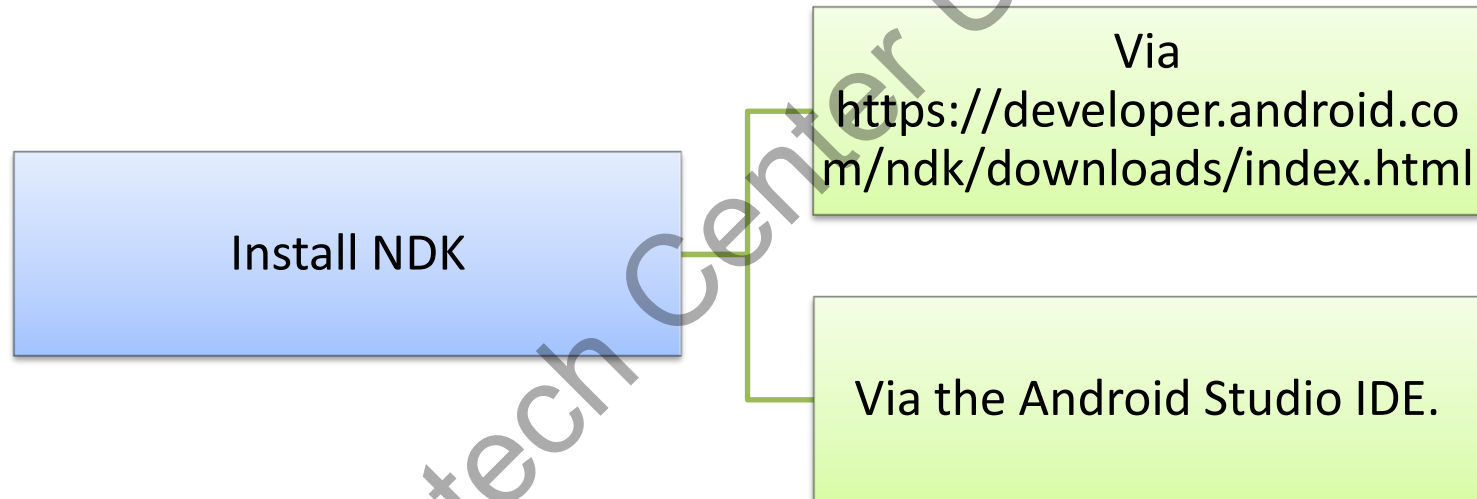
- 1 • Design the app.
- 2 • Decide the Java and native code components that need to be used in the app.
- 3 • Create an Android app project.
- 4 • For a native-only app, declare the **NativeActivity** class in **AndroidManifest.xml**.
- 5 • Create an **Android.mk** file.
- 6 • Add the native source code 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.

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.

For Aptech Center Use Only

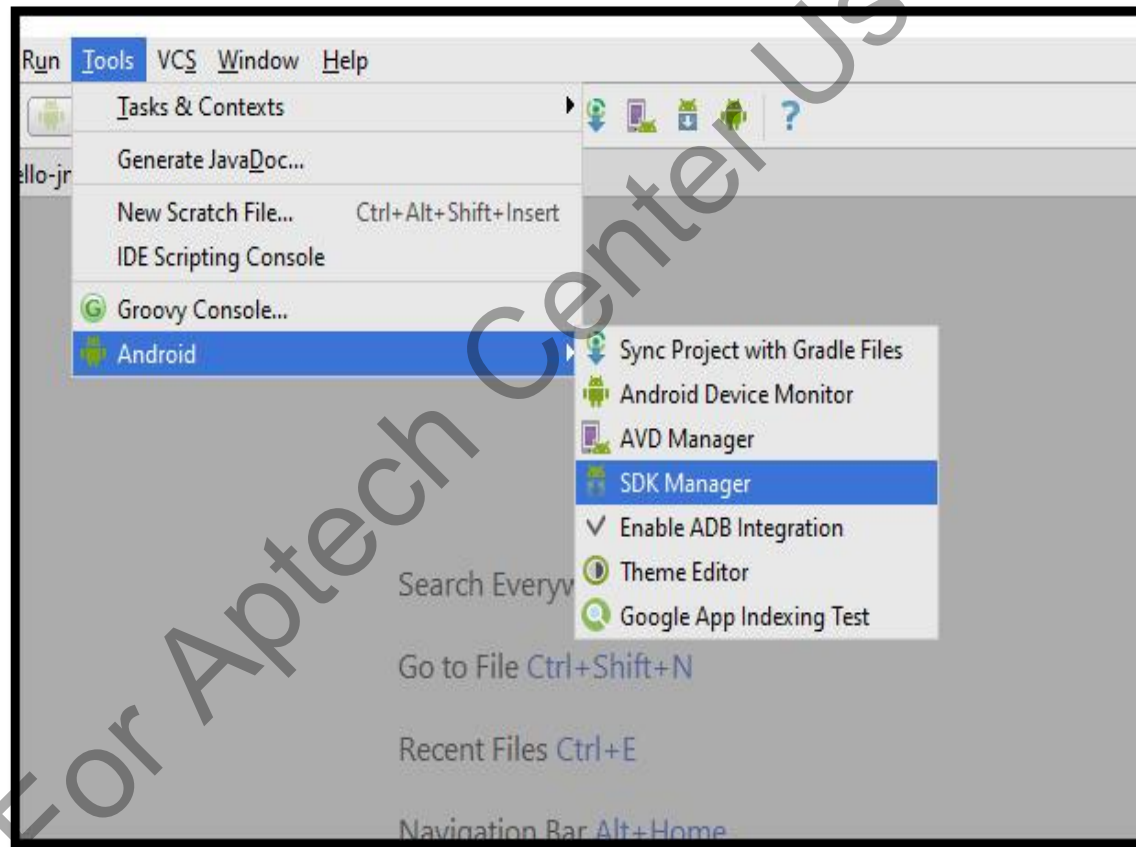
Installing the NDK 1-3



Installing the NDK 2-3

Alternate approach to install NDK

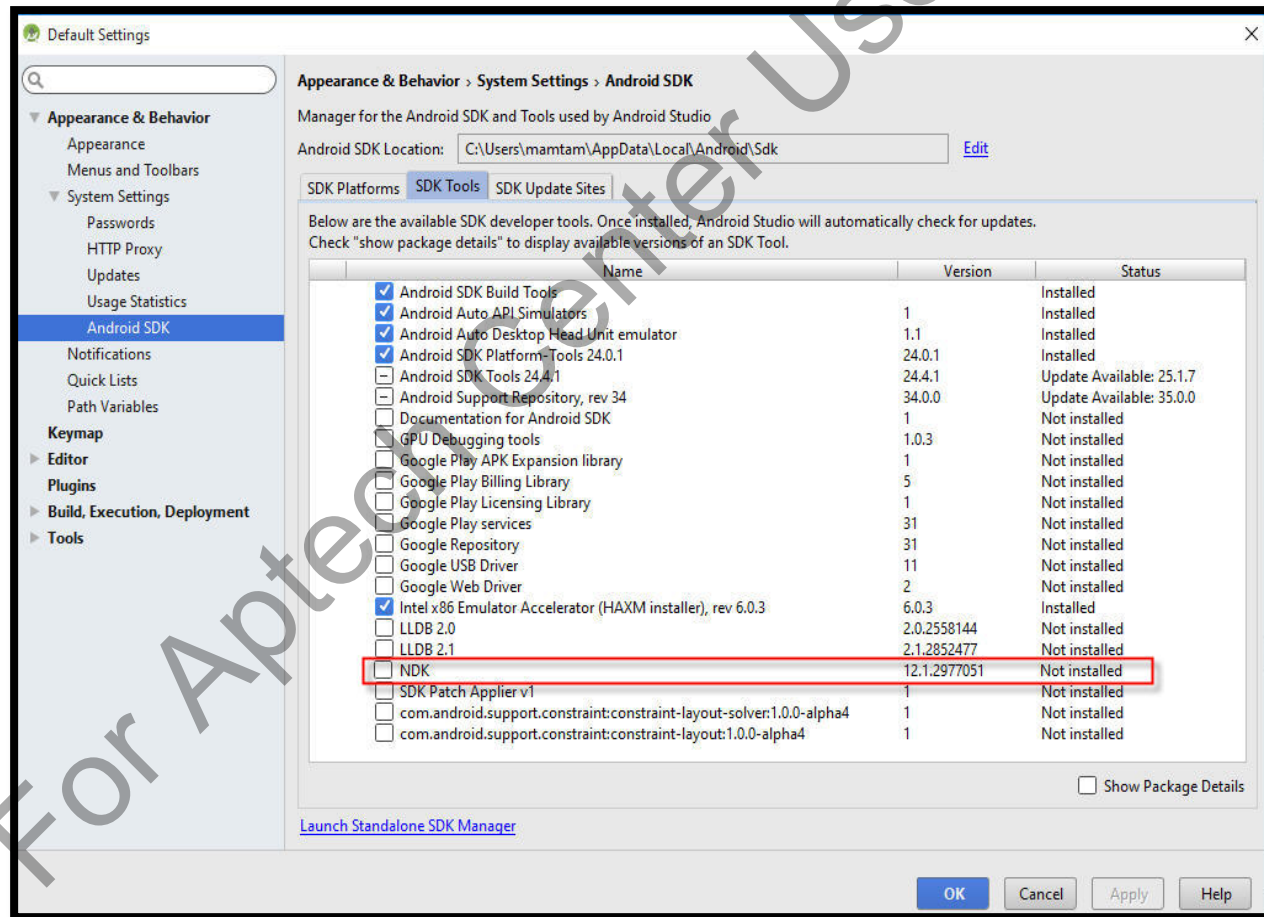
Following figure shows how to launch Android Studio:



Installing the NDK 3-3

Alternate approach to install NDK

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



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++.

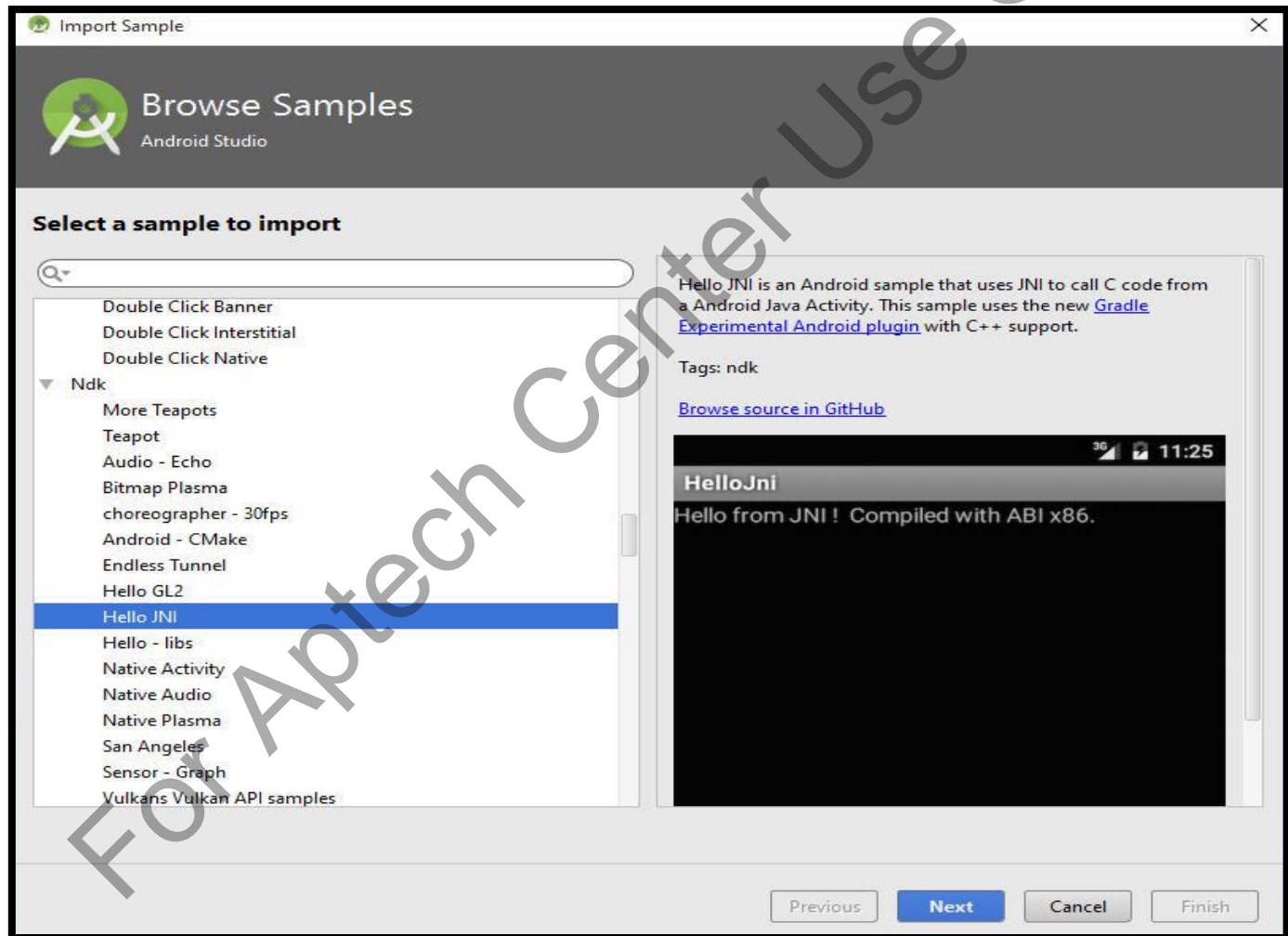
Creating and Testing Sample App 2-8

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



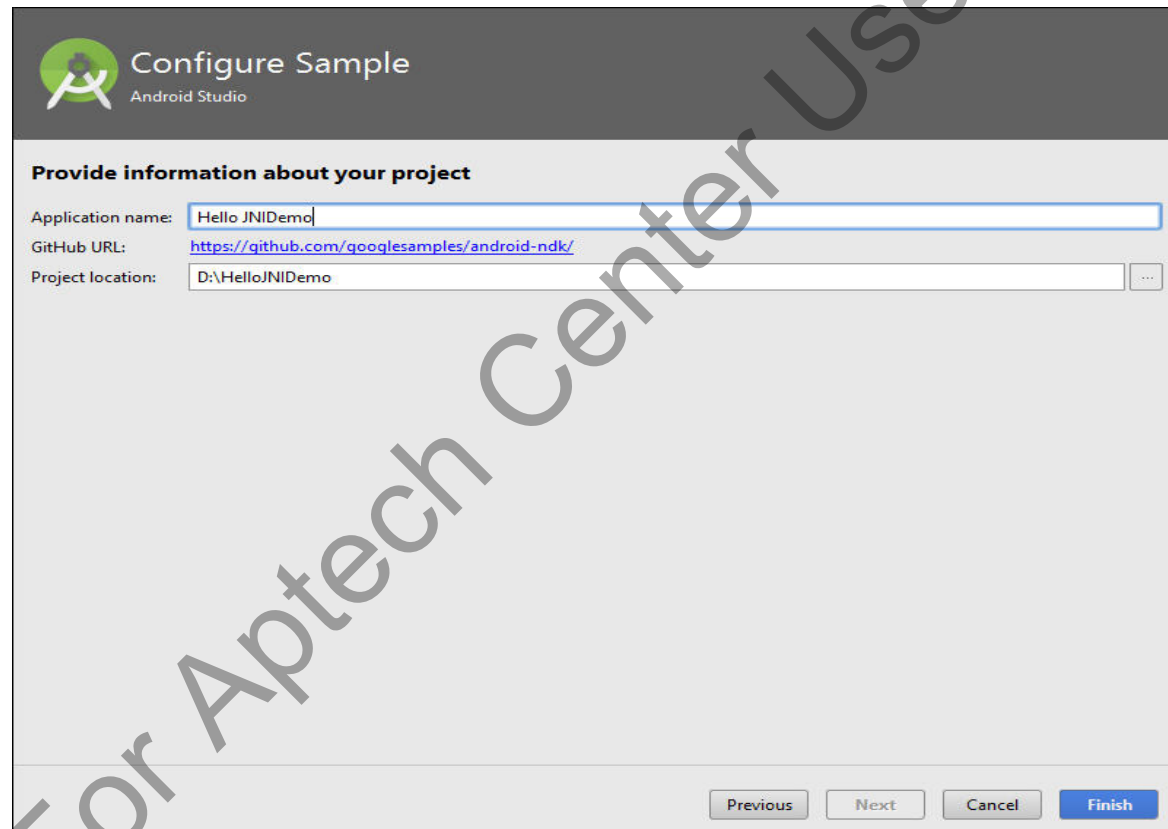
Creating and Testing Sample App 3-8

❑ Following figure shows how to select Hello JNI:



Creating and Testing Sample App 4-8

❑ Following figure shows how to specify application name:



The screenshot shows the 'Configure Sample' dialog in Android Studio. The dialog has a dark header with the Android Studio logo and the text 'Configure Sample' and 'Android Studio'. Below the header, the section 'Provide information about your project' is visible. It contains three input fields: 'Application name' with the text 'Hello JNIDemo', 'GitHub URL' with the text 'https://github.com/googlesamples/android-ndk/', and 'Project location' with the text 'D:\HelloJNIDemo'. At the bottom of the dialog, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

Configure Sample
Android Studio

Provide information about your project

Application name: Hello JNIDemo

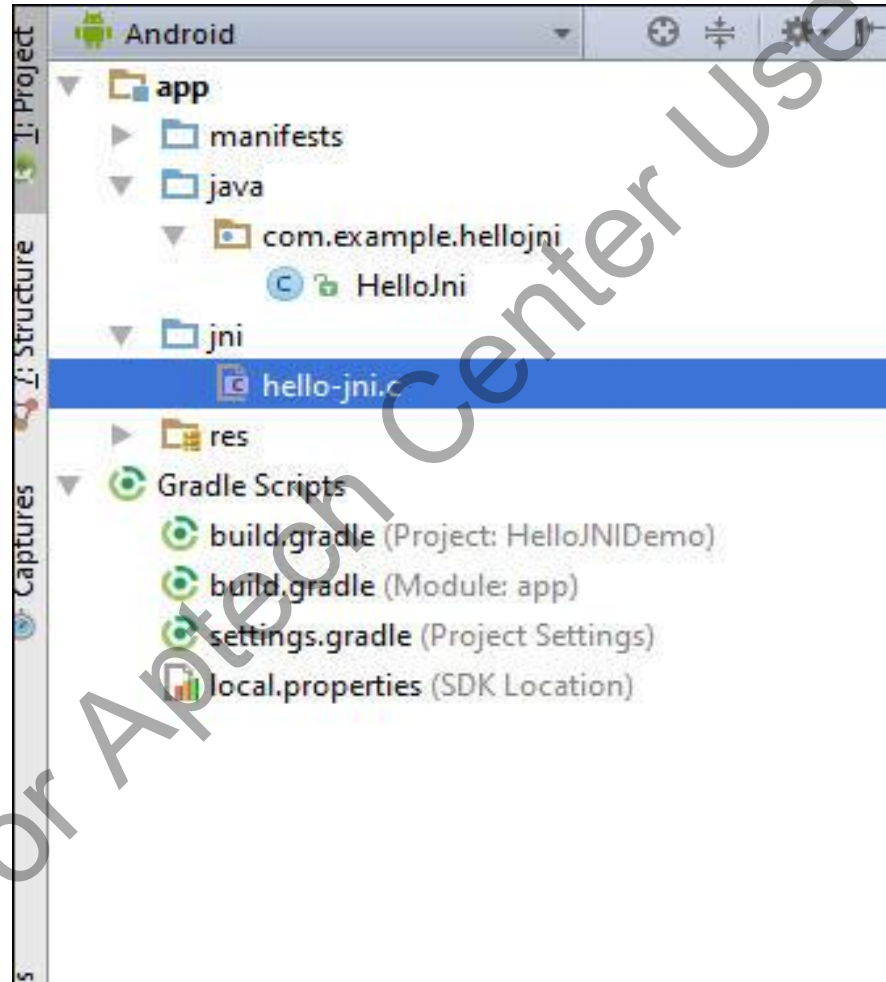
GitHub URL: <https://github.com/googlesamples/android-ndk/>

Project location: D:\HelloJNIDemo

Previous Next Cancel Finish

Creating and Testing Sample App 5-8

❑ Following figure displays the imported files:



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

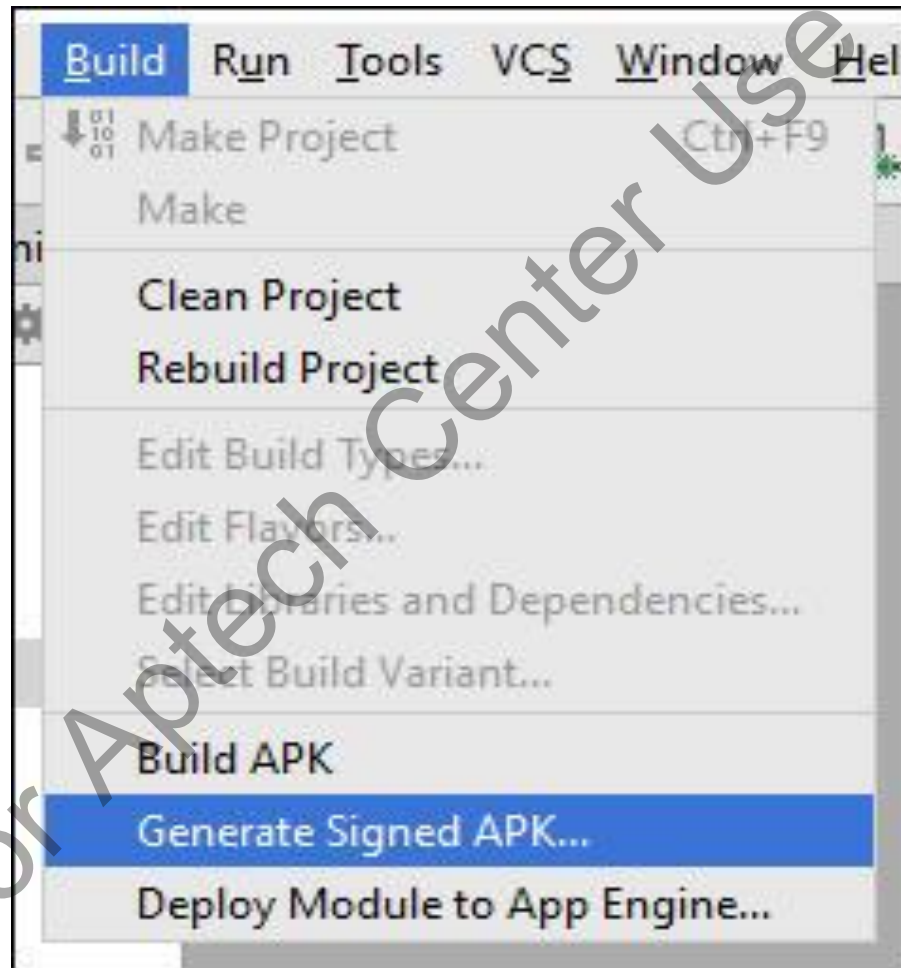
Enables Java applications to interact with native code.

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

For Aptech Center Use Only

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.