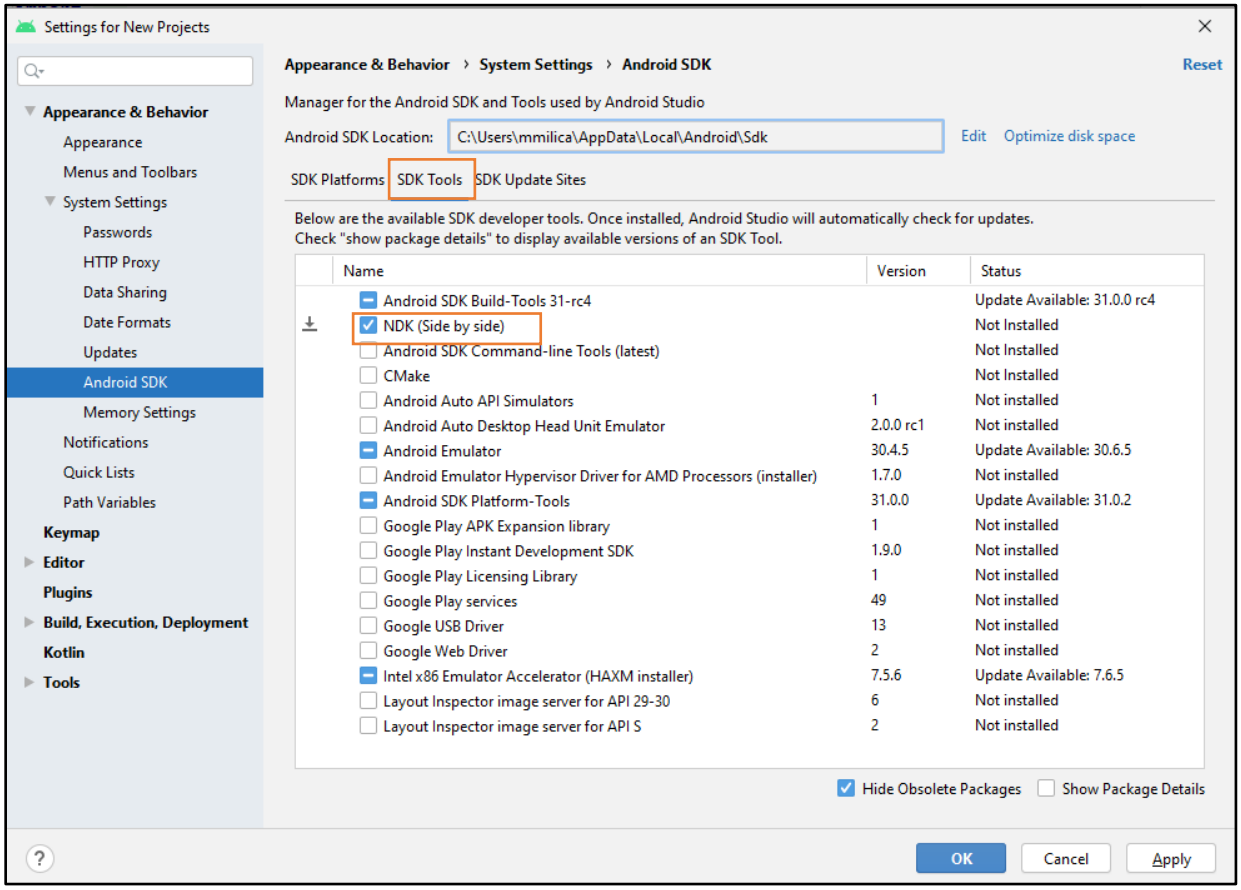
# JNI - Java Native Interface

JNI (Java Native Interface) omogućuje pozivanje nativnih (C/C++) funkcija (koje i dalje čine deo aplikacije) iz Java koda.

U nastavku su objašnjeni koraci konfiguracije projekta kako bi podržao JNI kao i primer povezivanja nativne funkcije sa Java kodom.

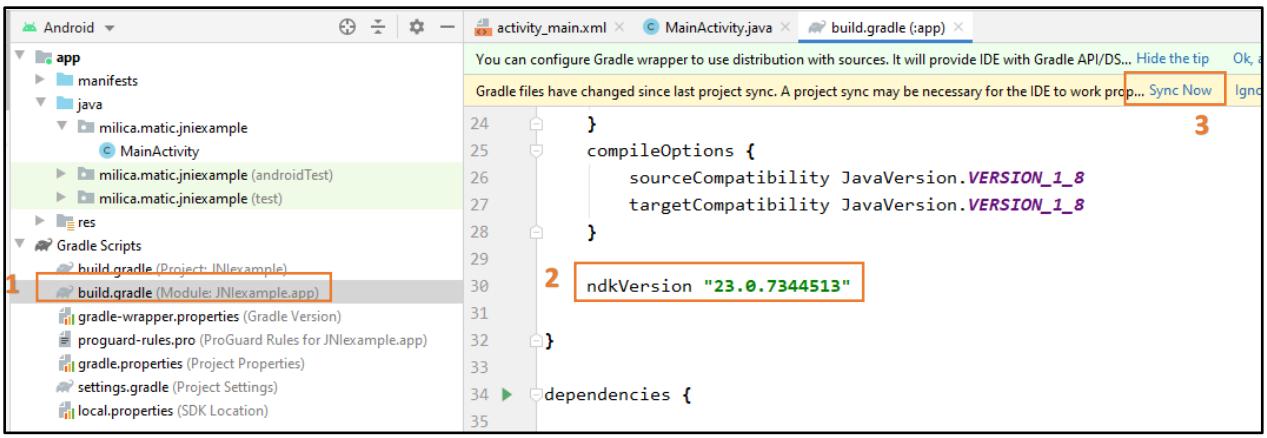
## Instalacija NDK (Native Development Kit)

1. U Android Studio-u odabrati opciju: **Tools -> SDK Manager -> SDK Tools -> NDK** kao što je prikazano na - Slika 1.




*Slika 1 - Instalacija NDK*

2. Proširiti Gradle skriptu (Slika 2 - 1) sa NDK verzijom (Slika 2 - 2). NDK verzija se može pronaći u SDK manager prozoru Android Studio-a ili na putanji *SDK_path*/ndk/broj_verzije. Nakon toga sinhronizovati projekat (*sync now* opcija) (Slika 2 - 3).




*Slika 2 - Podešavanje Gradle skripte*

# Primer poziva nativne funkcije iz Java koda

1. Kreirati novu Java klasu i u njoj definisati *static* blok u kome se učitava biblioteka koja će biti kreirana, kao i nativne metode koje će biti implementirane u JNI.

```
public class JNIexample {

    static {
        System.loadLibrary( libname: "MyLibrary");
    }

    public native int increment(int x);
}
```

*Slika 3 - JNI; Java klasa*

U ovom koraku je bitno primetiti dve stvari:

- Biblioteka koja će biti kreirana od C/C++ funkcija će imati naziv ***MyLibrary***
- Biblioteka će imati jednu metodu koja se zove **increment**, prima jedan parametar tipa **int** i povratna vrednost joj je tipa **int**.

2. Prevesti projekat.

3. Generisati .h datoteku. Posle verzije **9** Java ukida **javah** komandu i generisanje *header* datoteka se izvršava pokretanjem **javac** komande sa parametrom **-h**. Slede komande i njihovo objašnjenje za oba slučaja:
    - Java **pre** verzije 9 (koristi se **javah** komanda; ukoliko se pojavi greška da **javah** nije prepoznata kao komanda, potrebno ju je dodati kao Environment varijablu):

```
    javah -o app\src\main\jni\myJni.h -classpath
app\build\intermediates\javac\debug\classes package.name.JNIexample
```

- Putanja i naziv .h datoteke koja se kreira
- Direktorijum u kom se, nakon prevođenja, nalazi Java klasa u kojoj su definisane nativne metode (JNIexample)
- Naziv Java klase u kojoj smo definisali nativne metode

- Java **nakon** verzije 9 (koristi se komanda **javac** sa parametrom **-h**):

```
    javac -h app\src\main\jni
app\src\main\java\jniexample\JNIexample.java
```

- Putanja na kojoj će se kreirati zaglavlje
- Putanja do Java klase u kojoj su definisane nativne metode (JNIexample)

Poređenje metode definisane u Java klasi MyNDK.java i u generisanoj .h datoteci:

- `public native int increment(int x)`
- `JNIEXPORT jint JNICALL Java_jniexample_JNIexample_increment (JNIEnv*, jobject, jint)`
  - `JNIEnv*` - referenca pomoću koje se pristupa svim JNI funkcijama
  - `jobject` - odgovara ključnoj reči *this* u Javi
  - `jint` - parametar koji funkcija prihvata (`int x`)

4. Na osnovu generisanog zaglavlja kreirati novu c/cpp datoteku u cpp direktorijumu (File -> New C/C++ Source File). U ovoj datoteci je potrebno impelementirati nativnu metodu.

```
#include "jniexample_JNIexample.h"

JNIEXPORT jint JNICALL Java_jniexample_JNIexample_increment
  (JNIEnv * env, jobject jobj, jint x) {

  return ++x;

}
```

*Slika 4 - Implementacija c/cpp datoteke*

Lista podržanih JNI funkcija:

https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html

Lista podržanih JNI tipova:

https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/types.html

5. Kreirati make datoteke u cpp direktorijumu: File -> New File
   - **Android.mk**

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := MyLibrary
LOCAL_SRC_FILES := myJni.cpp
include $(BUILD_SHARED_LIBRARY)
```

LOCAL_MODULE - naziv biblioteke koja se kreira

LOCAL_SRC_FILES - naziv c/cpp datoteka od kojih se generiše biblioteka

BUILD_SHARED_LIBRARY - označava da će se generisati **dinamička** biblioteka

   - **Application.mk**

```
APP_ABI := all
```

APP_ABI - definiše za koju platformu se generiše biblioteka (moguće je eksplicitno navesti, npr x86, arm…)

6. Za prevođenje nativnog koda i generisanje biblioteke se koristi komanda **ndk-build**. U terminalu se pozicionirati u **jni** direktorijum i iz njega pozvati komandu **ndk-build:**

```
>> cd app\src\main\jni

>> C:\Users\user\AppData\Local\Android\Sdk\ndk\broj_verzije\ndk-build.cmd
```

Ukoliko je komanda uspešno izvršena, u teminalu se dobija ispis sličan ispisu na slici ispod:

```
[arm64-v8a] Compile++      : MyLibrary <= myJni.cpp
[arm64-v8a] SharedLibrary  : libMyLibrary.so
[arm64-v8a] Install        : libMyLibrary.so => libs/arm64-v8a/libMyLibrary.so
[x86_64] Compile++      : MyLibrary <= myJni.cpp
[x86_64] SharedLibrary  : libMyLibrary.so
[x86_64] Install        : libMyLibrary.so => libs/x86_64/libMyLibrary.so
[armeabi-v7a] Compile++ thumb: MyLibrary <= myJni.cpp
[armeabi-v7a] SharedLibrary  : libMyLibrary.so
[armeabi-v7a] Install        : libMyLibrary.so => libs/armeabi-v7a/libMyLibrary.so
[x86] Compile++      : MyLibrary <= myJni.cpp
[x86] SharedLibrary  : libMyLibrary.so
[x86] Install        : libMyLibrary.so => libs/x86/libMyLibrary.so
```

Posle uspešnog izvršavanja komande **ndk-build** generisaće se dinamičke biblioteke u *libs* direktorijumu.

7. Kako bi se kreirana biblioteka uvezala sa Android projektom potrebno je izvršiti sledeće korake:
   - Desnim klikom na *cpp* direktorijum otvoriti novi meni
   - Odabrati opciju *Link C++ with Gradle*
   - Iz padajućeg menija odabrati opciju *ndk-build*
   - Proslediti putanju do Android.mk datoteke

8. Kada je biblioteka uspešno kreirana, moguće ju je koristiti iz Java klasa. U MainActivity instancirati objekat klase *JNIexample* i nad njim pozvati nativnu metodu *increment*:

```java
public class MainActivity extends AppCompatActivity {

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

        JNIexample jni = new JNIexample();
        int res = jni.increment( x: 5);
        Log.d( tag: "JNI_TAG", String.valueOf(res));
    }
}
```

*Slika 5 - Poziv nativne metode iz aktivnosti*

**Napomena**: Ukoliko se pojavi greška: **java.lang.UnsatisfiedLinkError** potrebno je uraditi sinhronizaciju projekta sa gradle: File -> Sync Project with Gradle

- Umesto CMake označiti ndk-build

- Obezbediti putanju do Android.mk datoteke: *path\to\project*\app\src\main\jni\Android.mk (path\to\project zameniti sa pravom apsolutnom putanjom do projekta).