안드로이드 OpenCV 사용하기

**NDK(Native Development Kit)란?**

C와 C++과 같은 네이티브 코드 언어를 사용하여 앱의 일부를 구현할 때 사용한다. NDK의 장점으로는 속도 향상, 유연한 프로그램 개발, OpenCV/OpenGL 관련 영상처리 라이브러리들을 안드로이드 프로젝트에서 사용할 수 있게 한다.

**JNI(Java Native Intefact)란?**

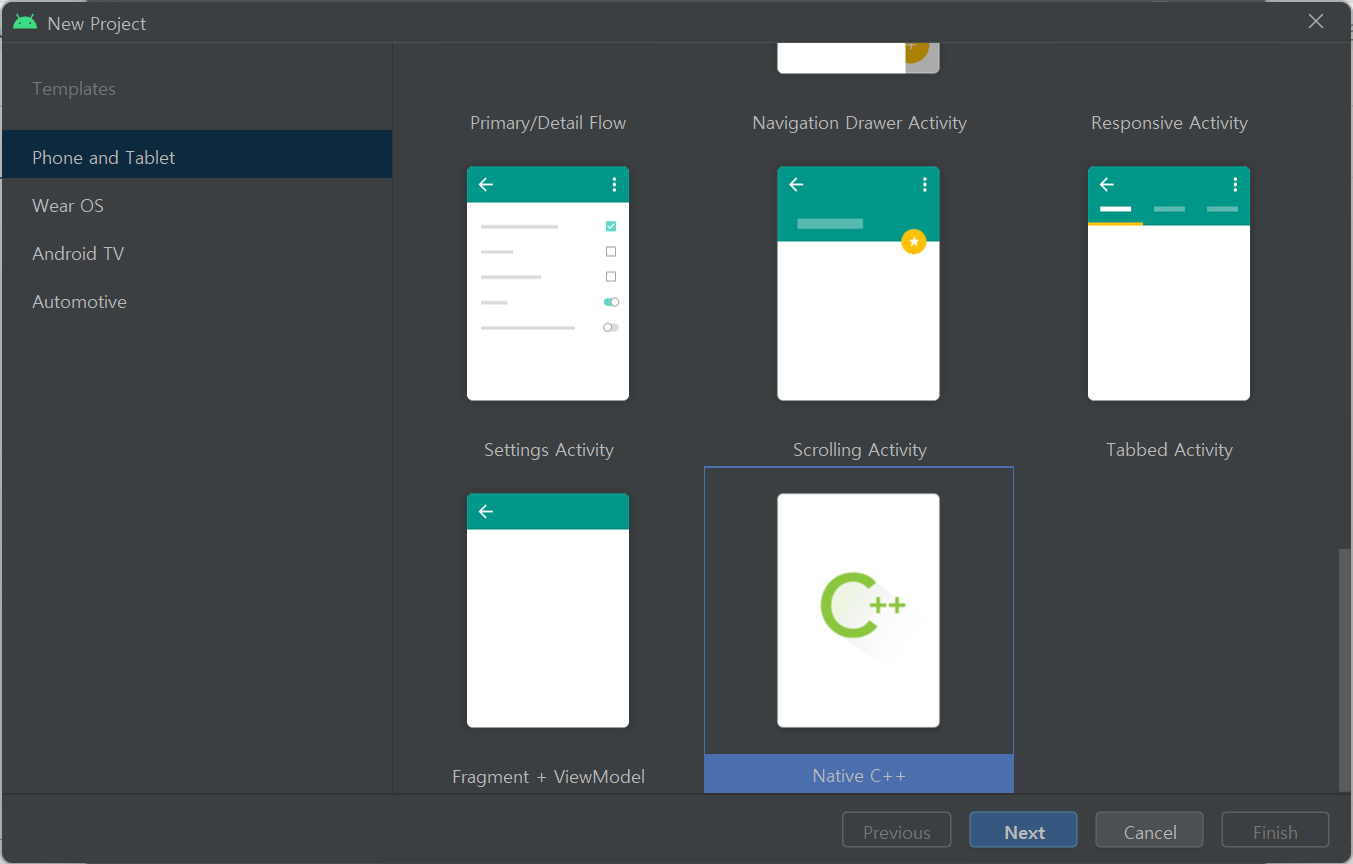
**Java와 C/C++ 라이브러리를 연동시켜주는 인터페이스이고, JNI를 사용하면 NDK를 사용할 수 있다.**

**1. NDK를 사용하기 위한 SDK Tool을 설치 (NDK, CMake)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**2. 프로젝트 생성 (JNI를 사용할 수 있는 Native C++ 프로젝트 선택)**

****

**3. OpenCV 라이브러리 다운로드 (**<https://github.com/opencv/opencv/releases>**)**

**텍스트, 모니터, 화면, 스크린샷이(가) 표시된 사진

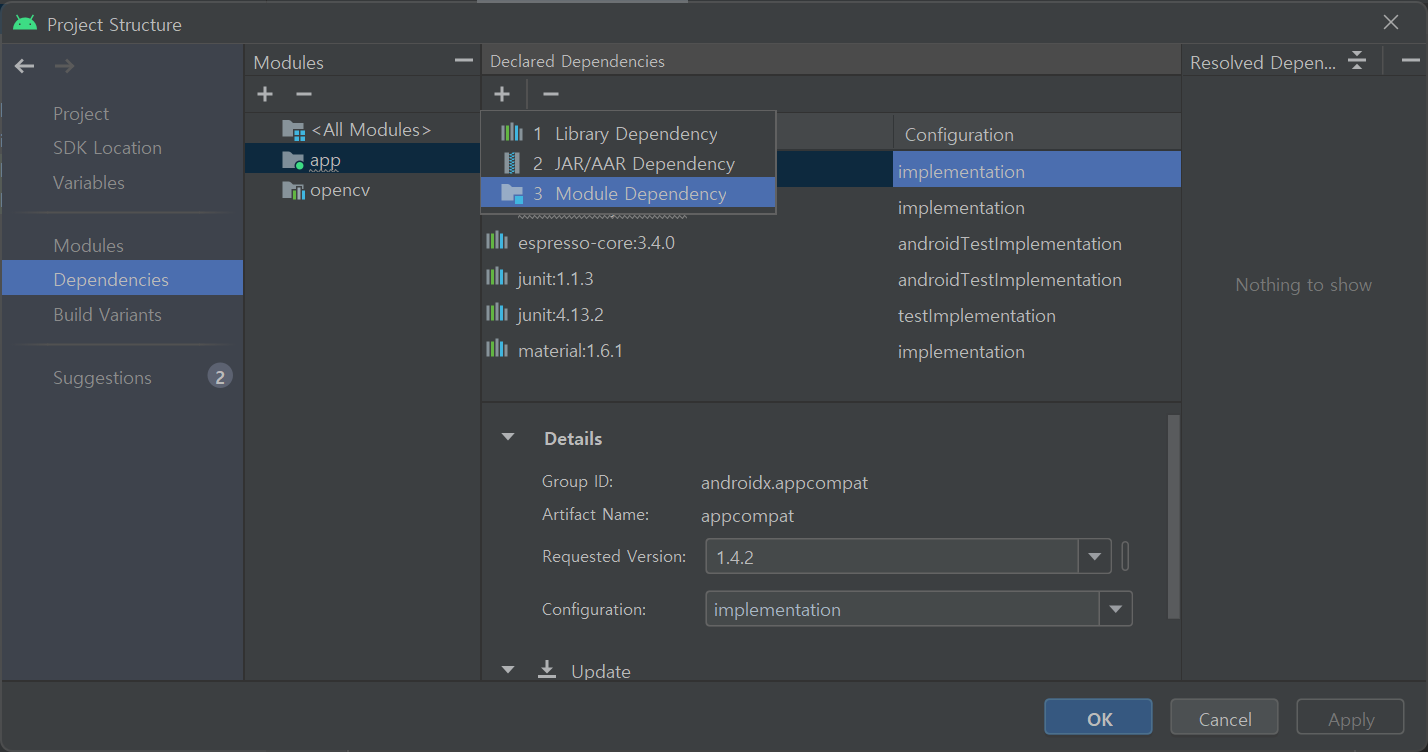
자동 생성된 설명**

**4. OpenCV 라이브러리 Import (File > New > Import Module from Source)**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**4. app 모듈에 Dependencies 설정 (File > Project Structure)**

****

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**5. opencv\build.gradlev 수정 (apply plugin: ‘kotiln-android’ 제거)**

**텍스트, 컴퓨터, 스크린샷, 모니터이(가) 표시된 사진

자동 생성된 설명**

**6. 아래와 같이 뜬다면 설정 성공 ^~^**

**텍스트, 장치, 게이지, 측정기이(가) 표시된 사진

자동 생성된 설명**

**OpenCV Java API를 사용하는 예시**

**public** **native** void detectEdgeJNI(long inputImage, long outputImage, int th1, int th2);

**static** {

System.loadLibrary("opencv\_java4");

System.loadLibrary("native-lib");

}

*/\*\*\**

*\* OpenCV Java API 사용 - 기본 이미지와 엣지 처리가된 이미지 뷰에 세팅*

*\*/*

**public** void detectEdge() {

Mat src = **new** Mat();

Utils.bitmapToMat(mInputImage, src);

Mat edge = **new** Mat();

Imgproc.Canny(src, edge, 50, 150);

Utils.matToBitmap(edge, mInputImage);

src.release();

edge.release();

mEdgeImageView.setImageBitmap(mInputImage);

}

*/\*\*\**

*\* JNI 사용 - 기본 이미지와 엣지 처리가된 이미지 뷰에 세팅*

*\*/*

**public** void detectEdgeUsingJNI() {

**if** (!mIsOpenCVReady) {

**return**;

}

Mat src = **new** Mat();

Utils.bitmapToMat(mInputImage, src);

mImageView.setImageBitmap(mOriginalImage);

Mat edge = **new** Mat();

detectEdgeJNI(src.getNativeObjAddr(), edge.getNativeObjAddr(), 50, 150);

Utils.matToBitmap(edge, mInputImage);

mEdgeImageView.setImageBitmap(mInputImage);

}