안드로이드의 전반적인 개념

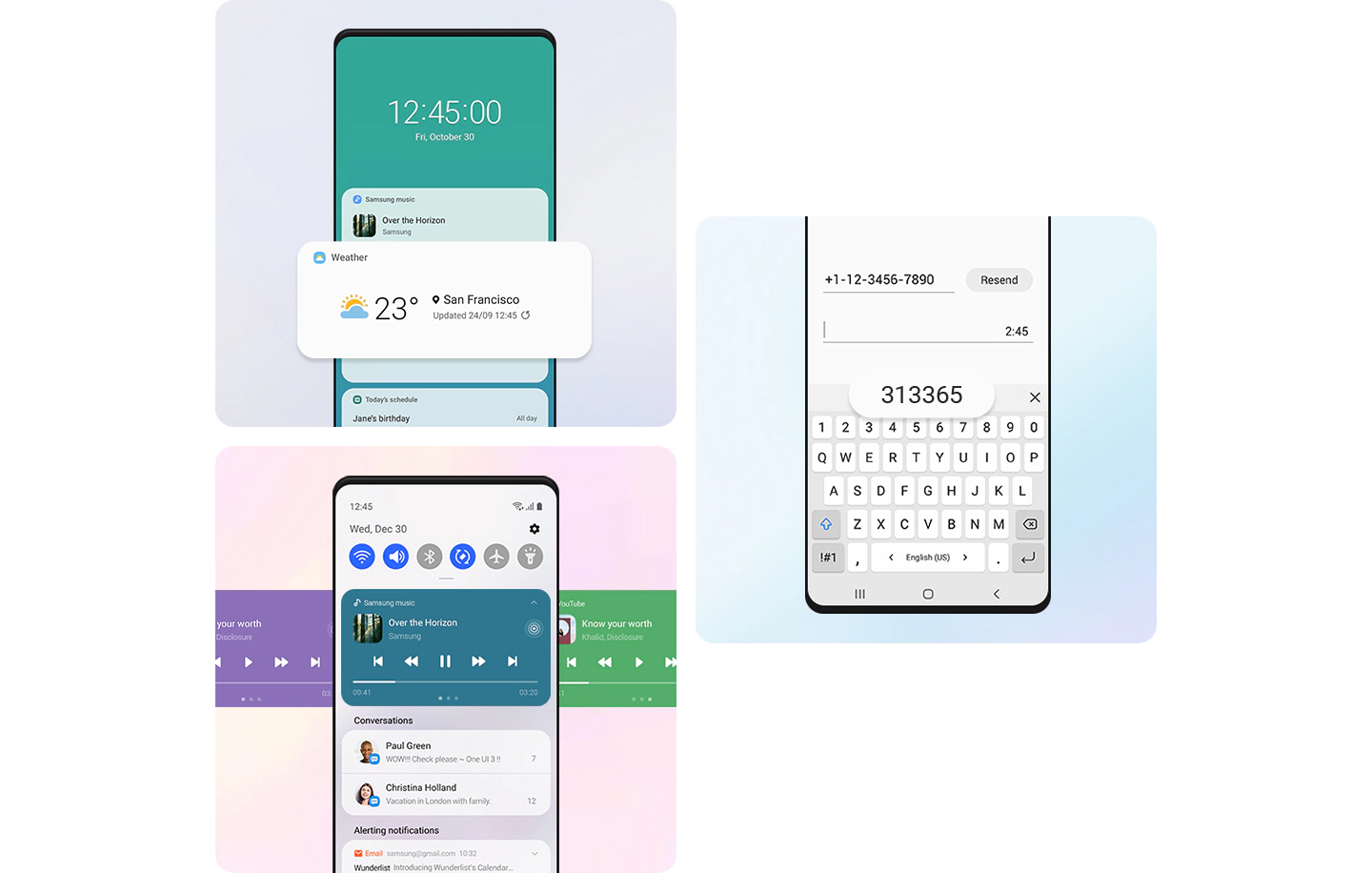
-동작방식에 대하여

12171742 최진우

|  |
| --- |
| 목차   1. 기본개념 2. 개발환경 구성방법 3. 동작방식    1. 커널 계층    2. 하드웨어 추상화 계층    3. Native Library / 안드로이드 런타임 계층    4. 애플리케이션 프레임워크 계층    5. 애플리케이션 계층 |

# 기본 개념

안드로이드는 터치스크린 디바이스를 겨냥하여 리눅스 커널과 그 외 오픈소스들을 활용해서 개발된 모바일 운영체제이다. 무료 오픈소스로서 Android Open Source Project로 알려진 Apache License로 배포되고 있고 스마트폰 제조사들은 이를 커스터마이징 하여 사용자들에게 배포하게 된다.



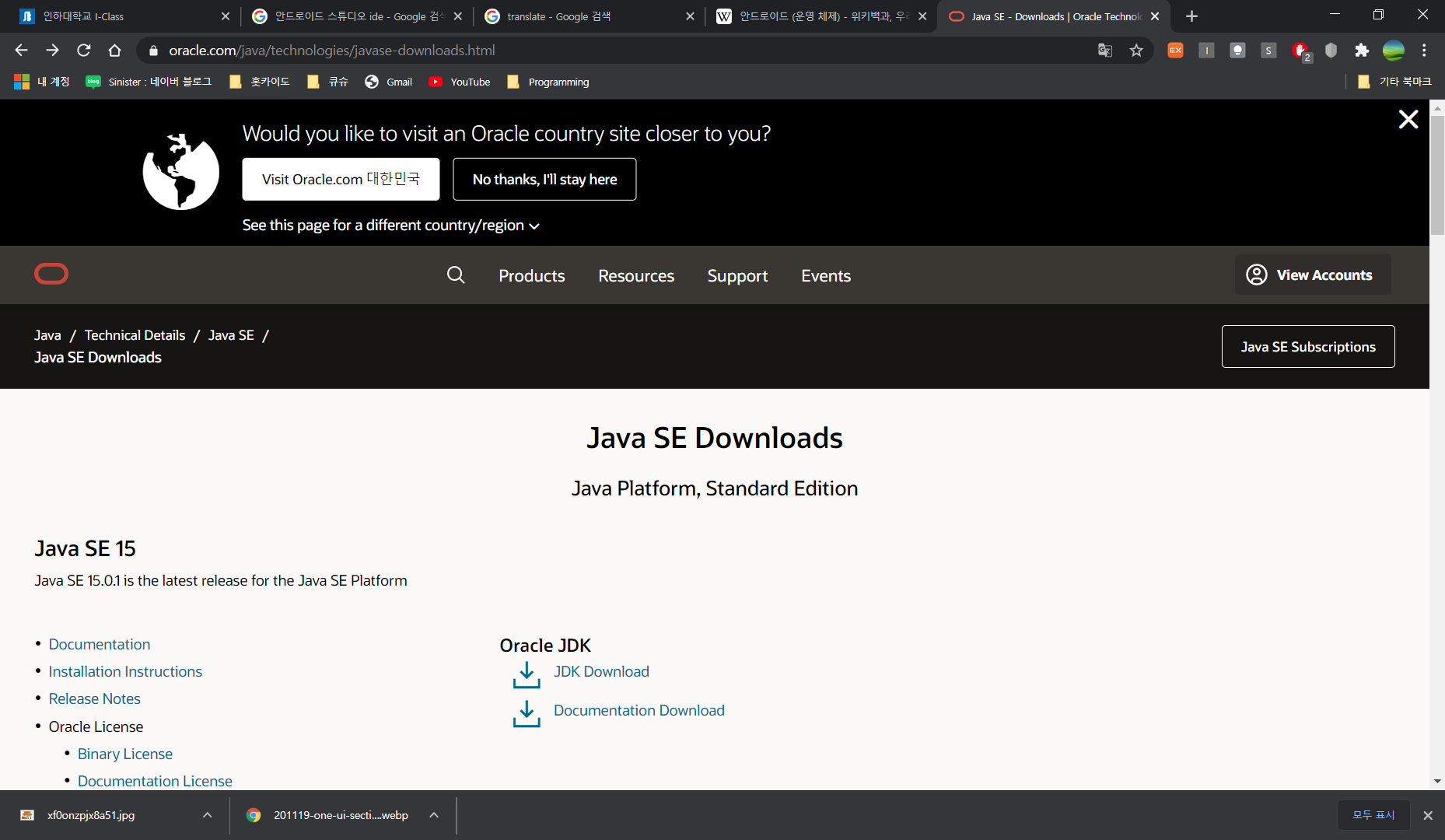
삼성의 ONE UI (출처: samsung.com) LG의 UX (출처: reddit.com)

위의 예시들이 앞서 언급한 제조사들에 의해 커스터마이징된 안드로이드이다. 이렇게 안드로이드 스마트폰에 적용되고 있는 소스코드만 보았지만 이외에도 콘솔게임장치, 디지털 카메라, pmp, 컴퓨터, 안드로이드tv, 스마트웨어 등등 많은 장치에 해당 용도에 특화된 사용자 인터페이스가 적용되어 탑재되고있다. 이러한 범용성에 힘입어 안드로이드는 2011년부터 최고로 많이 팔리는 스마트폰 os로 자리잡았으며,. 2013년부터는 태블릿시장에서도 1위os로 자리매김하였다. 안드로이드는 오픈소스이나 내부적으로는 구글 플레이, 크롬 등 구글생태계의 모바일 서비스를 사용하게 되어있는데 이 덕분에 약 70%이상의 안드로이드 탑재 제품에서 구글 서비스를 사용하고 있다. 안드로이드는 월간 보안 업데이트 및 연간 메이저 업데이트를 제공하는데 숫자 카운팅으로 메이저 업데이트 버전을 구분한다. 안드로이드 2.3부터 시작하여 현재는 안드로이드 11버전까지 나온 상태이다.

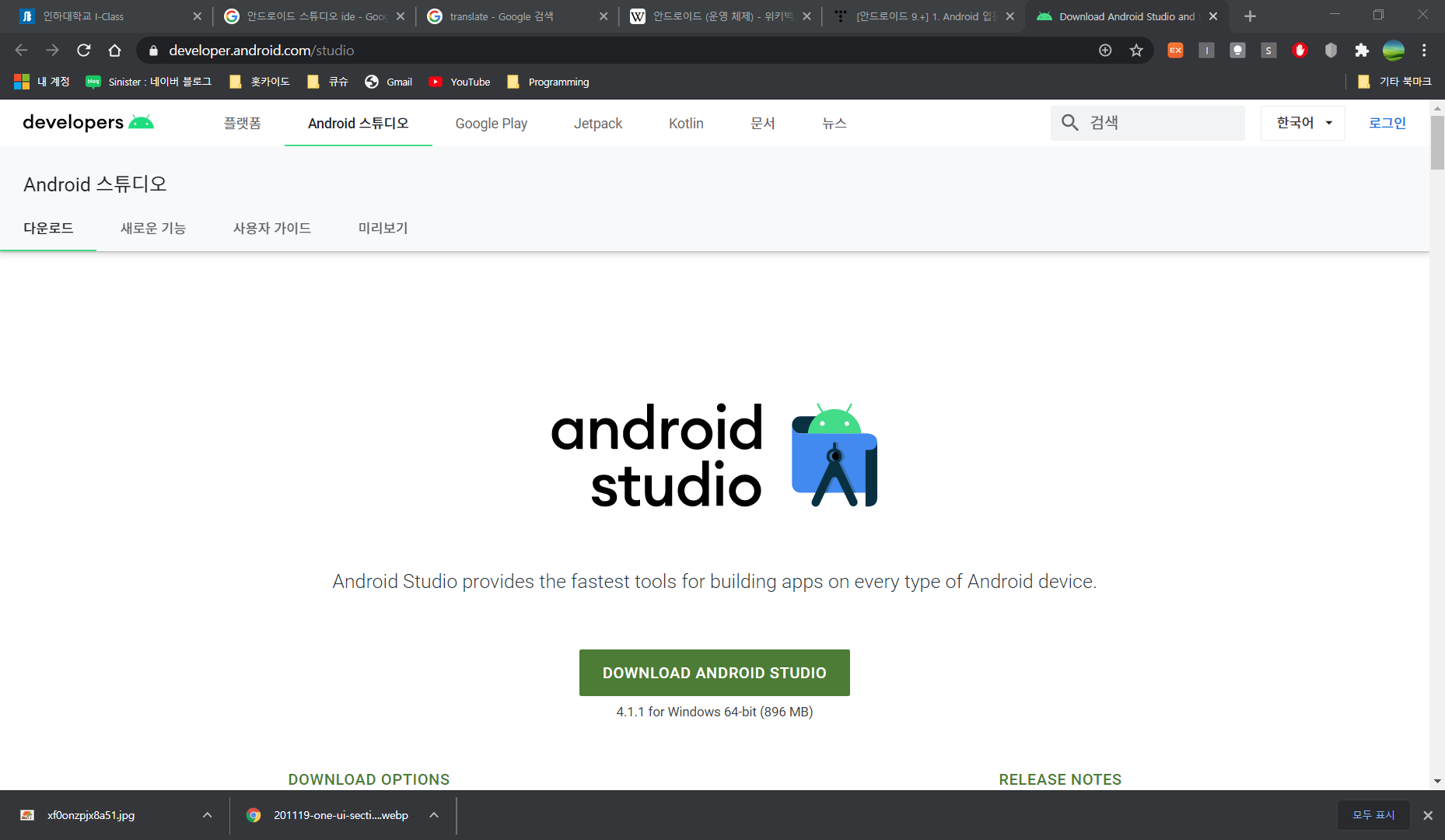
# 개발환경 구성방법

안드로이드는 Andriod Studio 라는 IDE를 통해 개발됩니다. 이때 java를 사용하여 개발하는데 이를 위해 jdk또한

필요하다. 아래는 간단한 개발환경 구성 방법을 단계별로 나타낸 것이다.

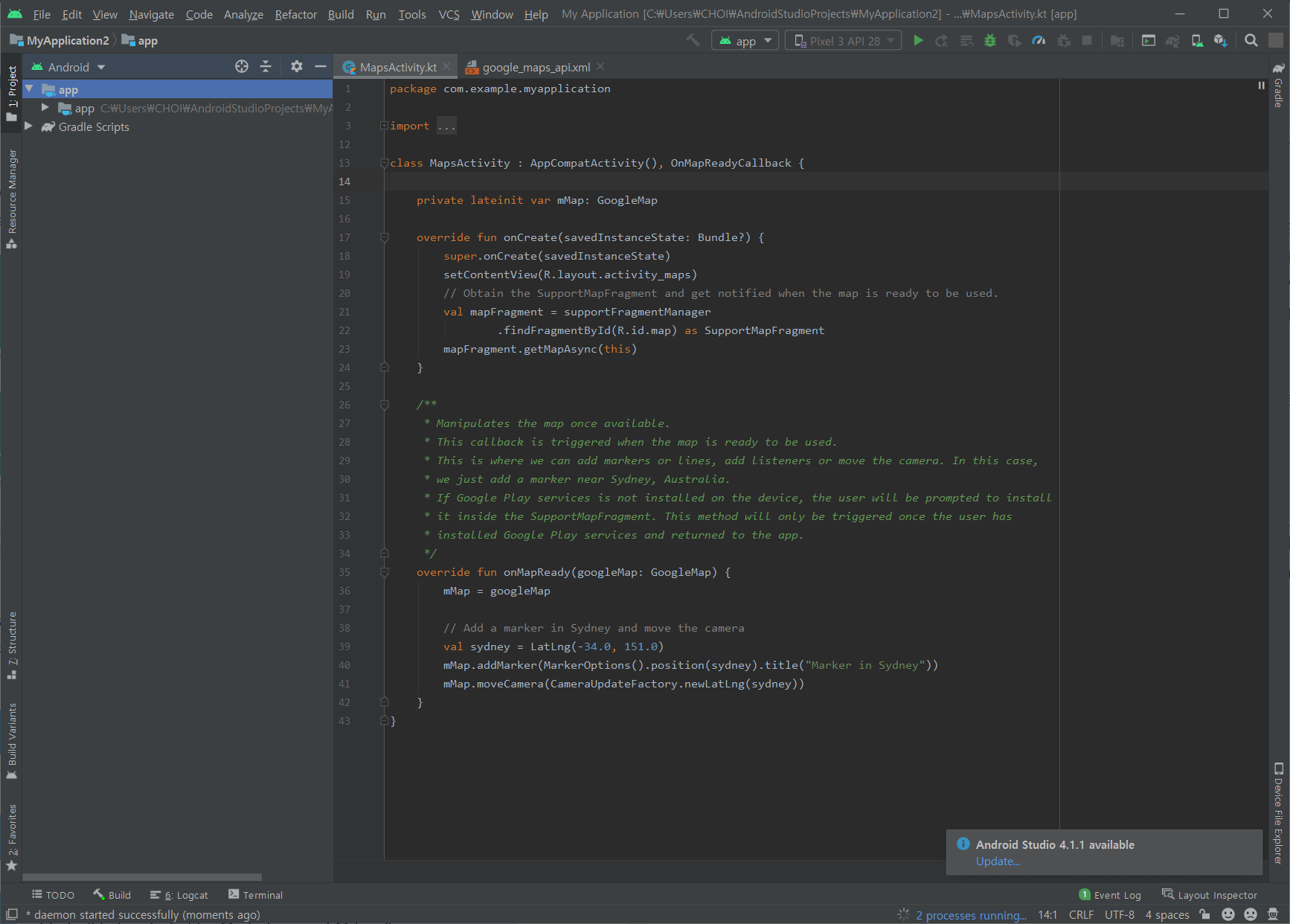


1. 앞서 설명한대로 java를 사용하기위해 oracle에서 jdk를 다운 후 설치한다.

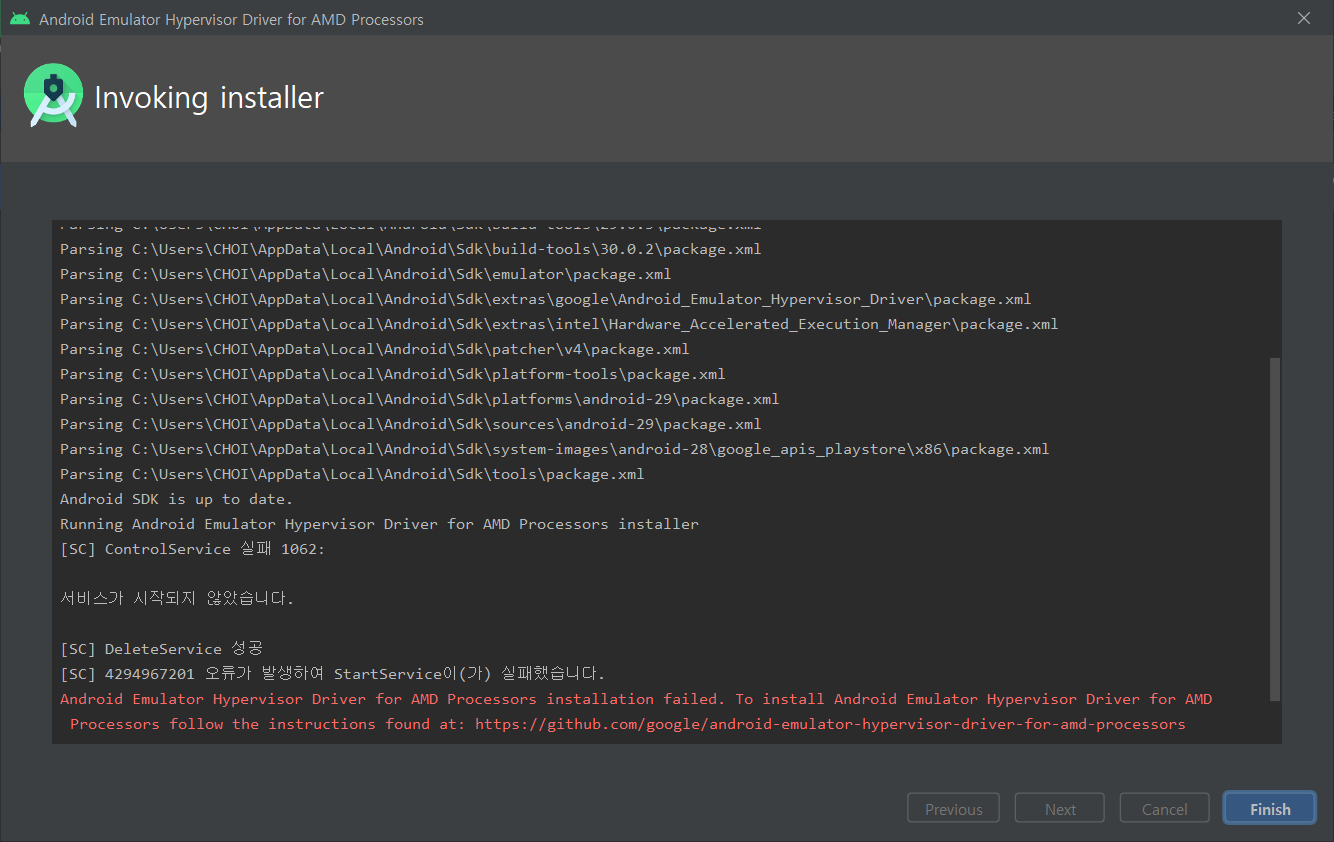


1. 이후 android studio를 (<https://developer.android.com/studio>)링크에서

다운 후 설치한다.



1. Android studio 실행 후 사용할 가상 디바이스 설정을 한다.



☞ 이때 AMD CPU의 경우 가상화 드라이버 오류가 발생하는데 bios에서 가상화 설정을 잡아 주어야한다. (참고 <https://friendcom.tistory.com/585>)

이로서 간단한 개발 환경 설정이 끝나게 된다.

# 동작방식

안드로이드 플랫폼은 다음과 같은 5계층으로 나뉘게 된다.

System Applications

phone

contacts

browser

etc

home

Java API FrameWork

Activity Manager

Content

Providers

Window

Manager

View

System

Notificaton

Manager

Package

Manager

Telephony

Manager

Resource

Manager

Location

Manager

XMPP

Service

Surface

Manager

Android Rumtime

Native C/C++ Libraries

FreeType

Media

Framework

SQLite

Core

Libraries

WebKit

OpenGL/ES

Android Runtime

(ART)

libc

SSL

SGL

HardWare Abstraction Layer (HAL)

Sensors

Camera

Bluetooth

Audio

etc

Linux Kernel

Power

Management

Flash Memory

Driver

Camera

Driver

Bluetooth

Driver

Keypad

Driver

WiFi

Driver

Binder (IPC)

Driver

Display

Driver

USB

Driver

Audio

Drivers

해당 다이어그램은 <https://developer.android.com/guide/platform>(안드로이드 공식 홈페이지)를 참조하였다. 이를 보면 리눅스 커널 계층, 네이티브 라이브러리 계층, 안드로이드 런타임 계층. 애플리케이션 프레임워크 계층, 애플리케이션 계층이 존재한다.

## 커널 계층

안드로이드의 최하위 계층에 위치한다. 이러한 커널은 리눅스 기반으로 안드로이드 플랫폼의 기초를 담당하고 있다고 할 수 있다. 커널은 하드웨어와 소프트웨어를 연결해주는 역할을 수행한다. 예를 들어 Android Runtime 계층의 ART는 하위수준의 메모리 관리 등의 기본기능에 리눅스 커널을 사용한다. 이러한 특성에 따라 리눅스를 지원하는 다른 기기들 또한 안드로이드로 구동이 잠재적으로 가능하다고 할 수 있다. 하지만 리눅스에다가 디바이스 구동에 필요한 몇가지 기능을 추가한 안드로이드는 리눅스와 약간씩 차이를 보인다. 이러한 차이는 커널 계층의 Power Management, Binder Driver에서 나타난다. Power Management는 항상 전원이 인가 되어있는 기존의 리눅스 기기들과 다르게 배터리기반으로 동작하는 안드로이드 기기들의 지원을 위해 새로 추가된 기능이다. 여기서는 대표적으로 두가지 기능이 있는데 앱 대기 버킷, 배터리 세이버이다. 첫째로 앱 대기 버킷은 안드로이드 파이(9) 버전부터 추가된 배터리 관리기능으로 앱의 리소스 요청에 대해 우선순위를 부여하는 기능이다. 이러한 순위는 좌측의 그림과 같이 5가지의 버킷으로 나뉘는데 해당 어플리케이션이 특정 조건을 만족 할 시 5개의 버킷 중 하나의 버킷으로 배치된다. 다음은 해당 버킷으로 들어갈 조건들을 보여준다.

Active bucket은 쉽게 말해 사용자가 해당 앱을 사용중인 경우이다. 이 버킷에서는 앱의 작업, 알람 등의 모든 작업에 대해 시스템의 제한을 걸지 않다. Working Set은 자주 실행되지만 현재는 활성상태가 아닌 경우에 해당된다. 앱이 간접적으로 실행될 때 또한 Working Set으로 승격된다. 이 버킷에서는 작업의 경우 최대 2시간 연기될 수 있고, 알람의 경우 최대 6분 연기될 수 있다. Frequent는 사용자가 운동시간에만 사용하는 운동추적 앱과 같이 정기적으로 사용되는 경우이다. 여기서는 작업이 최대 8시간 연기될 수 있고 알람 또한 최대 30분 연기될 수 있다. Rare는 사용자가 숙박할 때만 사용하는 숙박 앱과 같이 특정 조건에서만 실행되고 자주 사용되지 않는 경우에 해당된다. 이 때는 작업은 최대 24시간 연기되고 알람은 최대 2시간 연기될 수 있다. 또한 추가적으로 네트워크 연결 또한 최대 1일 제한될 수 있다. 마지막으로 never은 설치 이후 한 번도 실행하지 않은 경우이다. 이 경우 시스템에서 매우 엄격한 제한을 걸게 된다. 이러한 우선순위 버킷에는 시스템에 의해 앱이 동적으로 담기게 된다. 이때 제한은 기기가 배터리를 전원으로 사용 중 일 때만 적용되며 충전기를 사용 중 일 때는 시스템이 제한을 걸지 않는다. 둘째로 배터리 세이버는 기본적으로 시스템이 배터리관리를 위해 어플을 특정상황에서 제한하는 정책으로 순정 안드로이드인 AOSP에서는 대표적으로 다음과 같은 4가지의 전원 제한 정책을 가지고 있다.

* 시스템은 앱이 유휴 상태가 되는 것을 기다리는 것 보다 적극적으로 앱을 대기모드로 만들어준다.
* 백그라운드 실행을 막는 것은 API 레벨에 상관없이 모든 앱에 적용된다.
* 화면이 꺼지면 위치 서비스를 비활성화하게 만들 수 있다.
* 백그라운드 앱은 네트워크 접근 권한이 없다.

이러한 조건들을 통해 배터리를 사용하는 기기들의 배터리소모를 최적화하고 동작을 원할하게 만들 수 있다. 이제 Binder Driver(이하 바인더)에 대해 알아보자. 바인더는 기본적으로 IPC(네트워크상 각 프로세스들이 통신하는 형태 \*Inter Process Communication)도구다. 하지만 안드로이드에서는 이런 IPC의 한 종류인 RPC(Remote Procedure Call)를 바인더로 사용한다. 여기서 리눅스 메모리 공간에 대해 이야기해야 한다. 안드로이드 프로세스는 해당 프로세스 고유의 주소에서 실행되고 이렇게 여러 프로세스들은 독립된 주소공간에서 각자 별개로 동작하게 된다. 이러한 주소공간은 사용자 공간과 커널 공간으로 나뉘게 된다. 사용자공간에서는 사용자코드와 관련 라이브러리가, 커널 공간에서는 커널코드가 동작하게 된다. 이때 어떤 프로세스가 다른 프로세스로 데이터를 전달하기 위해서는 커널 공간을 통해야 한다. 왜냐하면 커널 공간은 프로세스들끼리 공유되기 때문이다. 이때 커널 입장에서 프로세스는 하나의 쓰레드 일 뿐이고 커널 공간 내부의 데이터 등은 전역변수와 같이 모두에게 공유된다. 이렇게 데이터를 커널 공간을 통해 전달 할 때 바인더가 사용된다. 예를 들어 카메라로 찍은 사진을 화면에 출력하는 경우, 다음과 같은 과정을 거친다. 이러한 과정이 안드로이드에서는 단순히 메시지를 전달하는 것이 아니라 상대방 프로세스에 존재하는 함수까지 호출하는 바인더를 사용하여 이루어진다. 이를 RPC방식 또는 메커니즘이라고 한다. 이러한 방식을 사용하여 앞에서 본 카메라 예시를 진행하면 카메라를 동작하는 프로세스가 사진을 출력하는 프로세스의 사진 출력함수를 커널 공간을 통해 호출하여 사진을 출력하게 된다. 이러한

카메라 동작 프로세스

사진 출력 프로세스

커널 공간

동작이 Binder IPC Driver라는 추상화된 드라이버를 통해 커널 공간에서 이루어 지는 것이다. 이러한 방식은 하위계층인 커널 부분을 통해 데이터를 전달하기 때문에 데이터의 신뢰성과 보안성이 매우 우수하다. 이렇게 커널 계층에서 기본적인 리눅스 커널과 다른 부분을 일부 살펴보았다. 이외의 Display driver, WiFi driver등은 기기에 맞는 드라이버를 커널 코드를 통해 구동시키는 리눅스의 기본적인 커널 부분과 같다.

## 하드웨어 추상화 계층

라