

모바일 프로그래밍

지정희

공학관 A동 1409-1호

450-3350

jhchi@konkuk.ac.kr

강의 개요

- 교재
 - 안드로이드 프로그래밍 관련 책 및 사이트
 - <https://developer.android.com>
 - <https://kotlinlang.org/>
- 평가
 - 중간고사(25%) / 프로젝트(30%)
 - 반드시 모든 시험 및 프로젝트에 응시 및 제출해야 학점이 부여됨
 - 과제(20%) / 랩실습(15%)
 - 출석(10%)
 - 대면 수업시 30분 지각은 결석 처리, 지각 3번은 결석 1회와 동일, 지각 1번도 감점됨
- 참고사항
 - 과제는 반드시 본인 스스로 해결할 것

Android 소개

Android 란?

- [위키백과]의 정의

- 안드로이드는 휴대전화를 비롯한 **휴대용 장치를 위한 운영체제와 미들웨어, 사용자 인터페이스 그리고 표준 응용 프로그램**(웹 브라우저, 이메일 클라이언트, 단문 메시지 서비스(SMS), 멀티미디어 메시지 서비스(MMS) 등)을 포함하고 있는 **소프트웨어 스택이자 모바일 운영체제**이다.

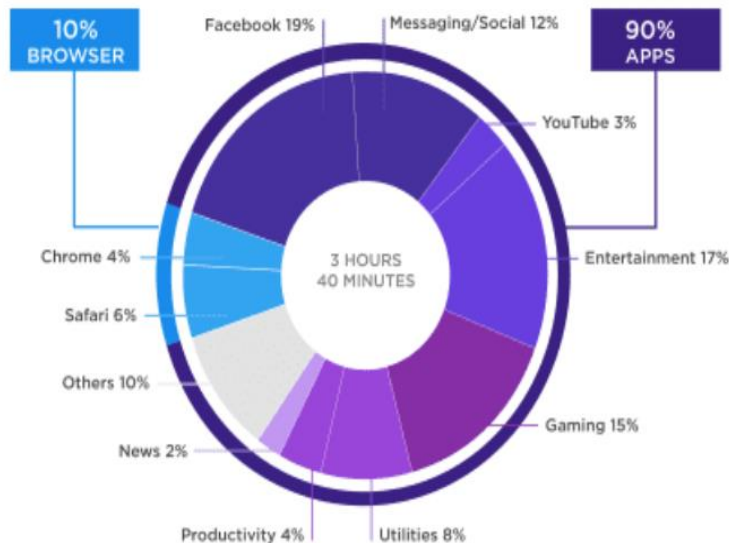


- [나무위키]의 정의

- 안드로이드는 **리눅스 커널**을 기반으로 구글에서 제작한 스마트폰과 같은 플랫폼의 **모바일 운영체제와 미들웨어 및 중요 애플리케이션이 포함된 소프트웨어 집합**이다. 구글은 새로운 운영체제의 버전 공개와 동시에 소스를 공개하고 있다. 이렇게 공개된 소스를 **AOSP(Android Open Source Project)** 라고 한다.

왜 Android인가?

- 왜 웹사이트로 작성하지 않는가?
 - 보다 **일관성 있는** 사용자 환경을 제공하는 **우수한 UI**를 제공함
 - 웹 페이지보다 다양한 종류의 widgets/controls을 사용할 수 있음
 - 디바이스의 하드웨어 장치(예, camera, GPS, 센서 등)에 직접 액세스 할 수 있음
 - 사용자는 모바일 웹 브라우징보다 앱에 대한 선호도가 더 높음



<https://buildfire.com/mobile-e-commerce-statistics-data/>



<https://www.richestsoft.com/blog/7-reasons-mobile-apps-better-mobile-websites/>

Android의 발전

- 2001년 : 모바일 단말에 맞춘 검색 서비스 시작
- 2005년
 - 안드로이드 인수
 - Skia 인수 (모바일 단말용 2D 그래픽)
 - RegWireless 인수 (모바일 단말용 브라우저와 이메일 클라이언트)
- 2007년 11월
 - 안드로이드 SDK 출시, OHA (Open Handset Alliance)
 - OHA는 전세계 30여개의 기업들(구글, 퀄컴, 모토로라, 삼성전자, LG 전자 등)이 참여한 다국적 연합체
- 2008년 9월
 - 안드로이드 SDK v1.0 릴리즈
 - G1 단말기 출시

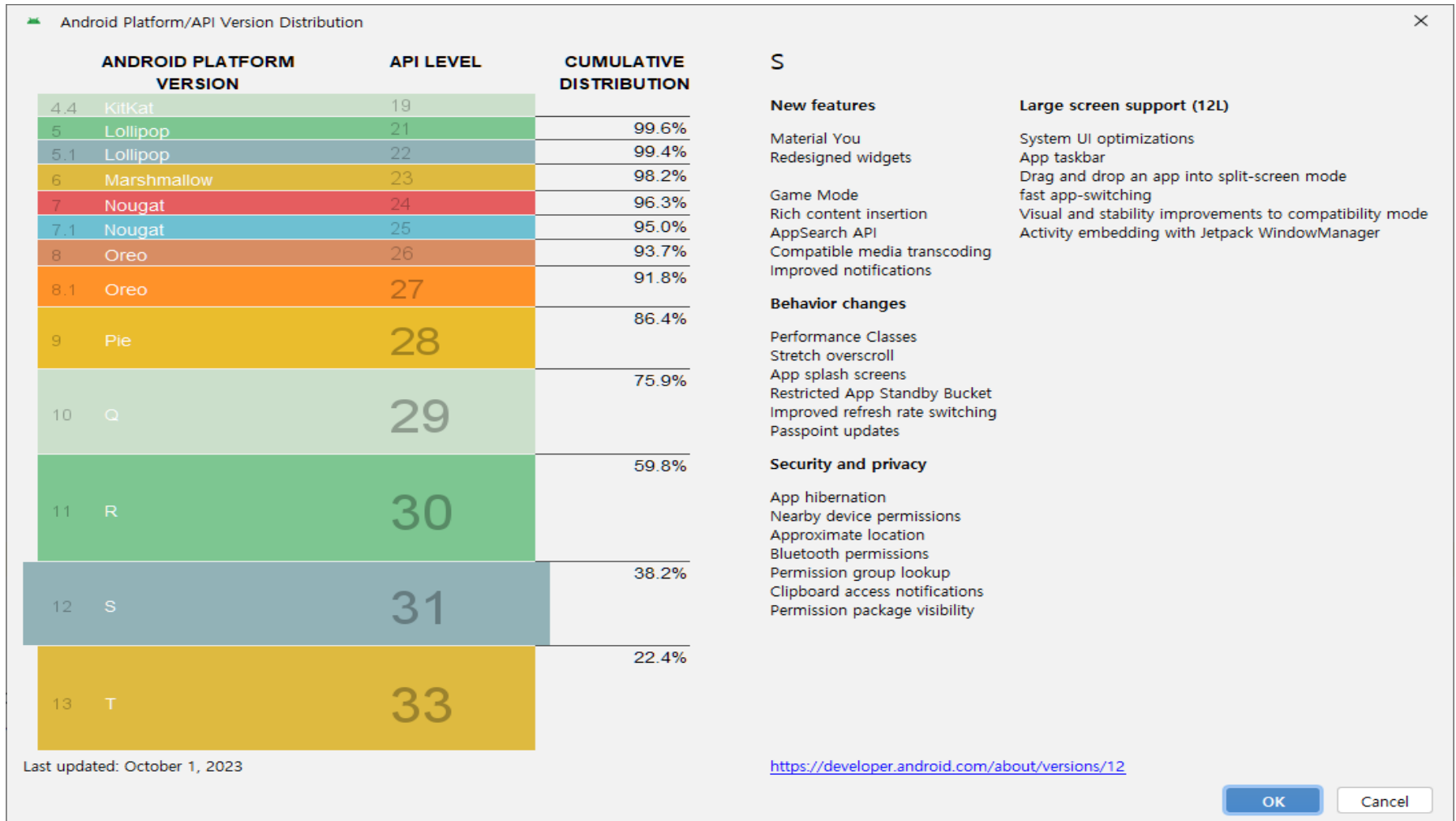
Android Version History

- https://en.wikipedia.org/wiki/Android_version_history

Version	API level	Date	Name
1.0-1.1	1,2	Sep 2008	None
1.5, 1.6	3, 4	Apr, Sep 2009	Cupcake, Donut
2.0-2.1	5,6,7	Oct 2009	Éclair
2.2	8	May 2010	Froyo
2.3	9,10	Dec 2010	Gingerbread
3.0	11,12,13	Feb 2011	Honeycomb
4.0	14,15	Oct 2011	Ice Cream Sandwich
4.1-4.3	16,17,18	Jun 2012	Jelly Bean
4.4	19,20	Sep 2013	KitKat
5.0-5.1	21-22	Jun 2014	Lollipop
6.0	23	Oct 2015	Marshmallow
7.0-7.1.2	24-25	Aug 2016	Nougat
8.0-8.1	26-27	Aug 2017	Oreo
9.0	28	Aug 2018	Pie
10.0	29	Sep 2019	Android 10 (Quince Tart)
11.0	30	Sep 2020	Android 11 (Red Velvet Cake)
12.0	31	Oct 2021	Android 12 (Snow Cone)
13.0	33	Aug 2022	Android 13 (Tiramisu)
14.0	34	Oct 2023	Android 14 (Upside Down Cake)

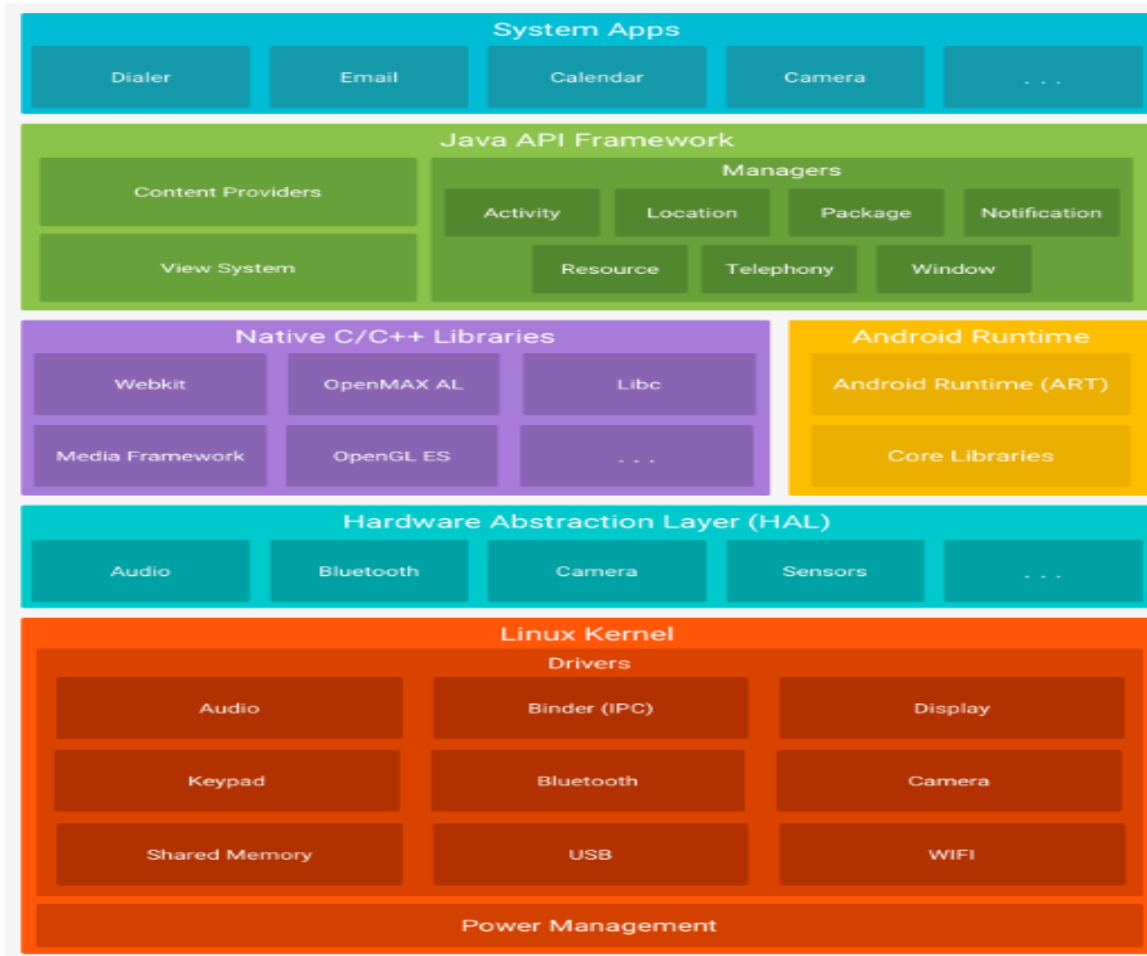
Android Version 배포 현황

- Android studio에서 확인 가능



Android Architecture

- 안드로이드 플랫폼의 구성

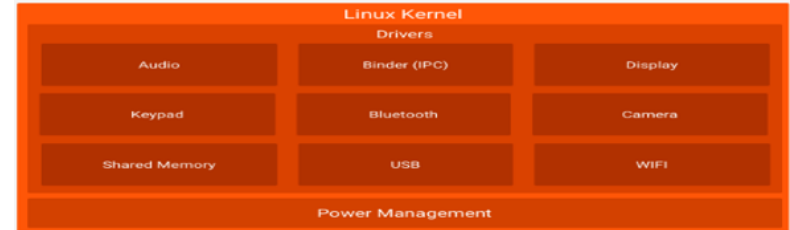


<https://developer.android.com/guide/platform/index.html?hl=ko>

Android Architecture

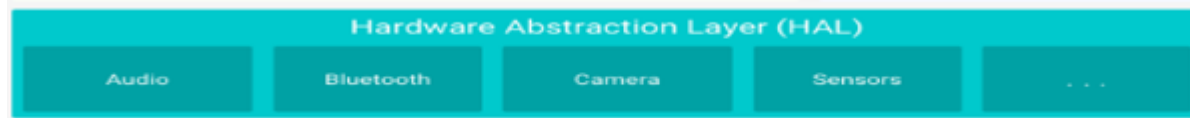
- 리눅스 커널

- 안드로이드는 리눅스 커널 상에 만들어져 있음
- 하드웨어 구동하는 기능 수행
 - 프로세스, 메모리, 전원, 네트워크, 디바이스 드라이버, 보안 등의 핵심적인 시스템 서비스를 지원함
- 표준 리눅스 유틸리티를 모두 제공하지는 않음
- 안드로이드 커널도 공개되어 있음
 - <https://android.googlesource.com/kernel/common/>



- HAL (Hardware Abstraction Layer)

- 자바 API 프레임워크에서 하드웨어 기능을 이용하는 표준 인터페이스 제공
 - 자바 API 프레임워크에서 하드웨어 기기(카메라, 블루투스 등)를 이용하기 위한 코드가 실행되면 내부적으로 HAL의 라이브러리 모듈이 로딩되어 처리됨



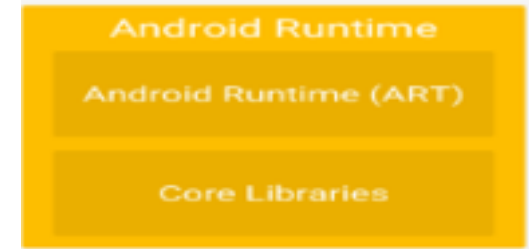
Android Architecture

- 네이티브 C/C++ 라이브러리
 - 장비의 전반적인 속도를 결정하는 중요한 요소임.
 - 안드로이드 시스템 구성요소와 서비스가 자바가 아니라 C/C++로 작성되어 있으며, 애플리케이션 프레임워크를 통해 사용할 수 있도록 구성되어 있음
 - 안드로이드 라이브러리 종류
 - 웹브라우저 및 웹렌더링 엔진 WebKit
 - 미디어 응용 개발 OpenMax AL
 - 임베디드용 c라이브러리 libc
 - 오디오, 비디오 재생을 위한 미디어 지원
 - 3차원 컴퓨터 그래픽스 OpenGL ES
 - 벡터 폰트 출력을 위한 FreeType
 - 데이터베이스 지원을 위한 SQLite



Android Architecture

- 안드로이드 런타임
 - 코어 라이브러리와 Android Runtime으로 구성되어 있음
 - 코어 라이브러리
 - 자바 라이브러리의 대부분 기능을 제공
 - ART (Android Runtime) – Android 5.0 (API 레벨 21) 이상
 - Dex 파일을 실행하여 저용량 메모리 기기에서 여러 가상 머신을 실행하도록 작성
 - Dex 파일 : 안드로이드 용으로 특별히 설계된 바이트 코드 형식
 - AOT (Ahead-Of-Time compile) : dex 파일을 설치할 때 한꺼번에 바이너리 형태로 만들어 사용
- * 달빅(Dalvik) 가상 머신 (자바 가상머신의 라이선스 문제)
 - 최소한의 메모리에 최적화된 DEX(Dalvik EXecute) 포맷을 사용하며 안드로이드 애플리케이션은 독립적 프로세스를 할당 받음
 - 자바의 가속화 기술인 JIT 기술을 채택하여 실행 속도 향상
 - JIT (Just In Time) : java의 바이트 코드를 앱이 실행할 때 바이너리 형태로 번역하여 캐쉬하여 사용



Android Architecture

- 애플리케이션 프레임워크 (Java API 프레임워크)
 - 안드로이드 API
 - 안드로이드에서 제공하는 애플리케이션도 애플리케이션 프레임워크의 API 기능을 기반으로 함
 - 응용 프로그램들은 하위의 커널이나 시스템 라이브러리를 직접적으로 호출할 수 없으며 API를 통해서 기능을 요청해야 함
- 애플리케이션 (응용프로그램)
 - 모든 응용프로그램은 애플리케이션 프레임워크의 API를 사용
 - 인터넷 브라우저, 바탕화면, 주소록 등 플랫폼과 함께 설치되는 애플릿들은 물론이고 마켓에서 다운받아 설치하는 게임, 유틸리티 등도 모두 이 수준에서 실행됨

안드로이드의 구성 컴포넌트

- 안드로이드 앱은 컴포넌트를 기반으로 함

- 안드로이드 컴포넌트란 안드로이드 시스템에서 생성하고, 관리하는 클래스
 - 일반 클래스는 개발자가 직접 생명주기를 관리하므로 컴포넌트 클래스가 아님
- 컴포넌트는 앱 내에서 독립적인 실행단위

- 인스턴화할 수 있는 4개의 컴포넌트

- 액티비티

- 사용자 인터페이스를 구성하는 기본 단위이며, 눈에 보이는 화면 하나가 액티비티이며, 여러 개의 뷰들로 구성
- 응용 프로그램은 필요한 만큼의 액티비티를 가질 수 있으며 그 중 어떤 것을 먼저 띄울지를 지정

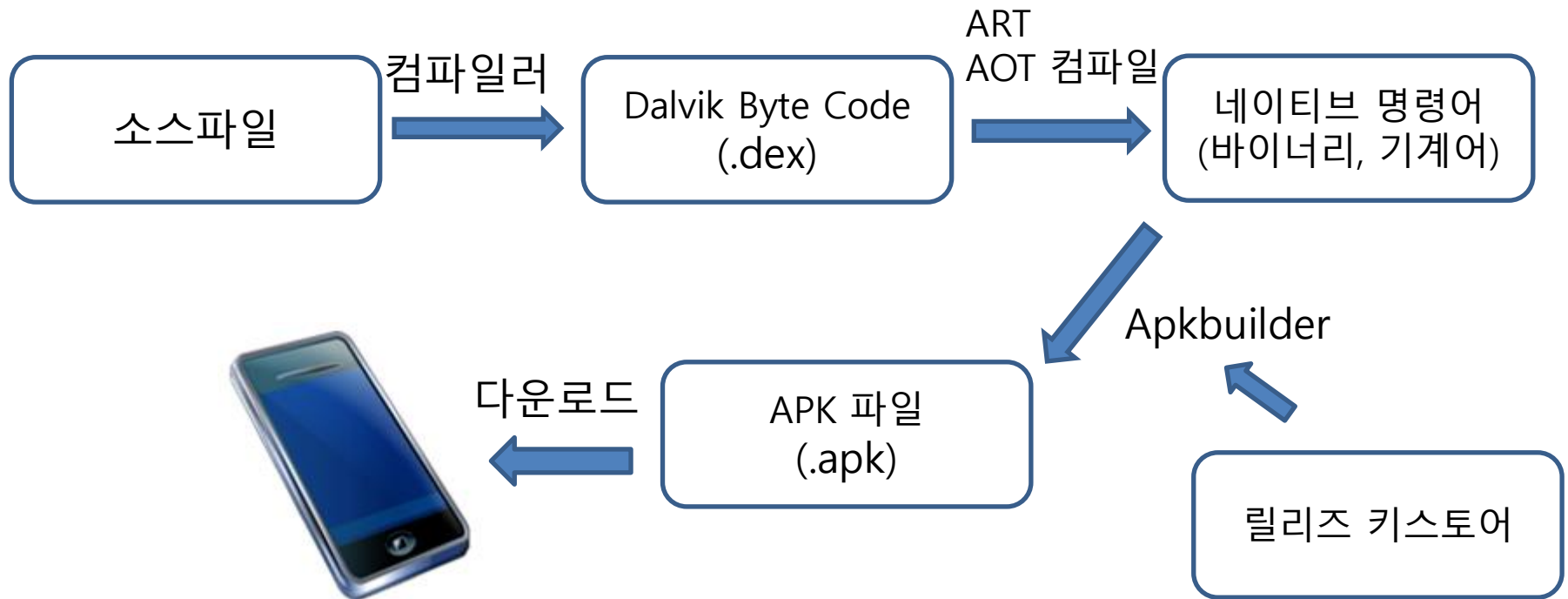
- 서비스

- UI가 없어 사용자 눈에 직접적으로 보이지 않으며 백그라운드에서 무한히 실행되는 컴포넌트
- 전형적인 예로 미디어 플레이어를 들 수 있는데 비활성화된 상태라도 노래는 계속 재생되어야 함
- UI가 없으므로 사용자의 명령을 받아들일 수 있는 액티비티와 연결해서 사용됨

안드로이드의 구성 컴포넌트

- 인스턴화할 수 있는 4개의 컴포넌트(계속)
 - 방송수신자(Broadcast Receiver)
 - 시스템으로부터 전달되는 방송을 대기하고 신호 전달 시 수신하는 역할
 - 예를 들어 배터리가 떨어졌다거나 사진을 찍었다거나 네트워크 전송이 완료 되었다는 등의 신호를 받음
 - 신호만 대기할 뿐 UI를 따로 가지지는 않으므로 방송 수신 시 방송의 의미를 해석하고 적절한 액티비티를 띄우는 역할
 - 콘텐츠 제공자(Content Provider)
 - 다른 응용 프로그램을 위해 자신의 데이터를 제공
 - 안드로이드는 보안이 엄격하여 다른 응용 프로그램의 데이터를 함부로 액세스 하지 못함
 - 응용 프로그램 간에 데이터를 공유할 수 있는 합법적인 유일한 장치가 바로 콘텐츠 제공자

실행 단계



- AAB(Android app bundle)
 - 2018년 구글 IO에서 발표한 새로운 안드로이드 앱 배포 파일
 - Play 스토어에 올리면 사용자 기기에 맞게 최적화된 APK를 대신 만들어 줌

개발환경 구축하기

시스템 요구사항

- Android Studio 시스템 요구사항
 - 윈도우 기준
 - 64비트 Microsoft® Windows® 8/10/11
 - x86_64 CPU 아키텍처.
 - 2세대 Intel Core 이상 또는 Windows 하이퍼바이저를 지원하는 AMD CPU
 - 8GB RAM 이상 / **가상 기기 사용할 경우 16GB RAM 이상**
 - 최소 8GB의 사용 가능한 디스크 공간(IDE + Android SDK + Android Emulator)
 - 1280x800 이상의 화면 해상도

안드로이드 개발 관련 프로그램


- 설치 프로그램
 - **Android Studio** (<http://developer.android.com/>)
 - IntelliJ 기반의 통합 개발 툴
 - Android SDK Tool
 - Gradle Build System
 - OpenJDK 내장
 - **Option : JDK** (<http://www.oracle.com>)
 - 시스템 고급설정에서 path에 경로 추가
 - 예) `c:\java\jdk\bin`
 - JAVA_HOME 변수 생성 및 경로 추가
 - 예) `c:\java\jdk`
- **주의 사항 : 사용자 계정에 한글이 들어가면 안됨**
 - 설치되는 경로상에 한글이 들어가면 안됨


Android Studio

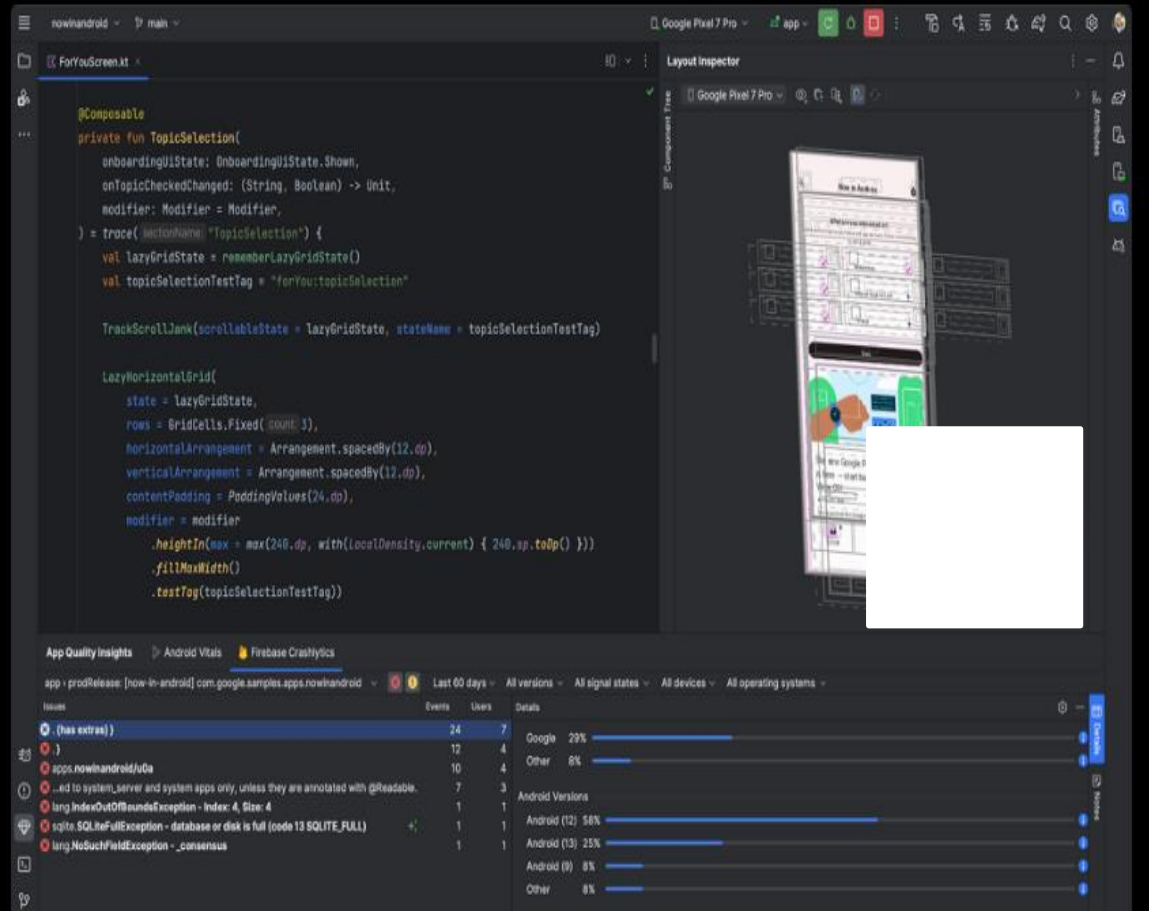
- Android Studio (<http://developer.android.com/>)

Android Studio

Android 앱 개발을 위한 공식 통합 개발 환경 (IDE)을 다운로드하세요.

Android 스튜디오 Hedgehog 다운로드 

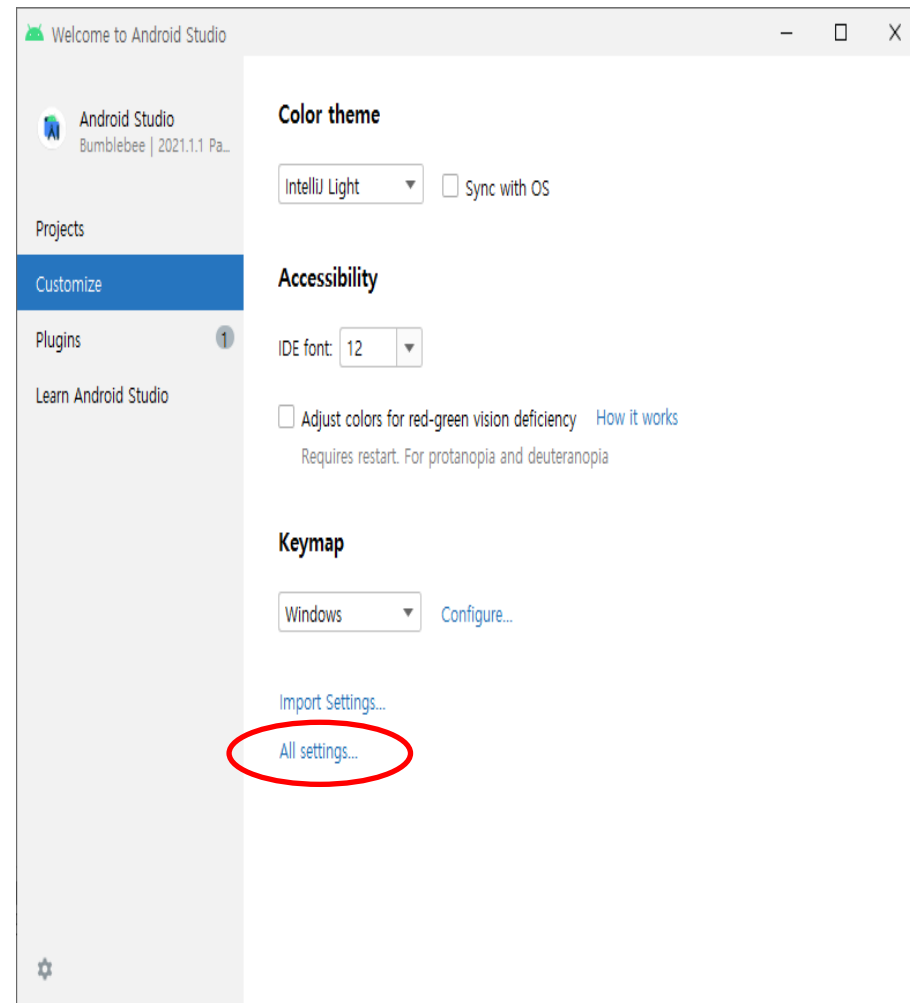
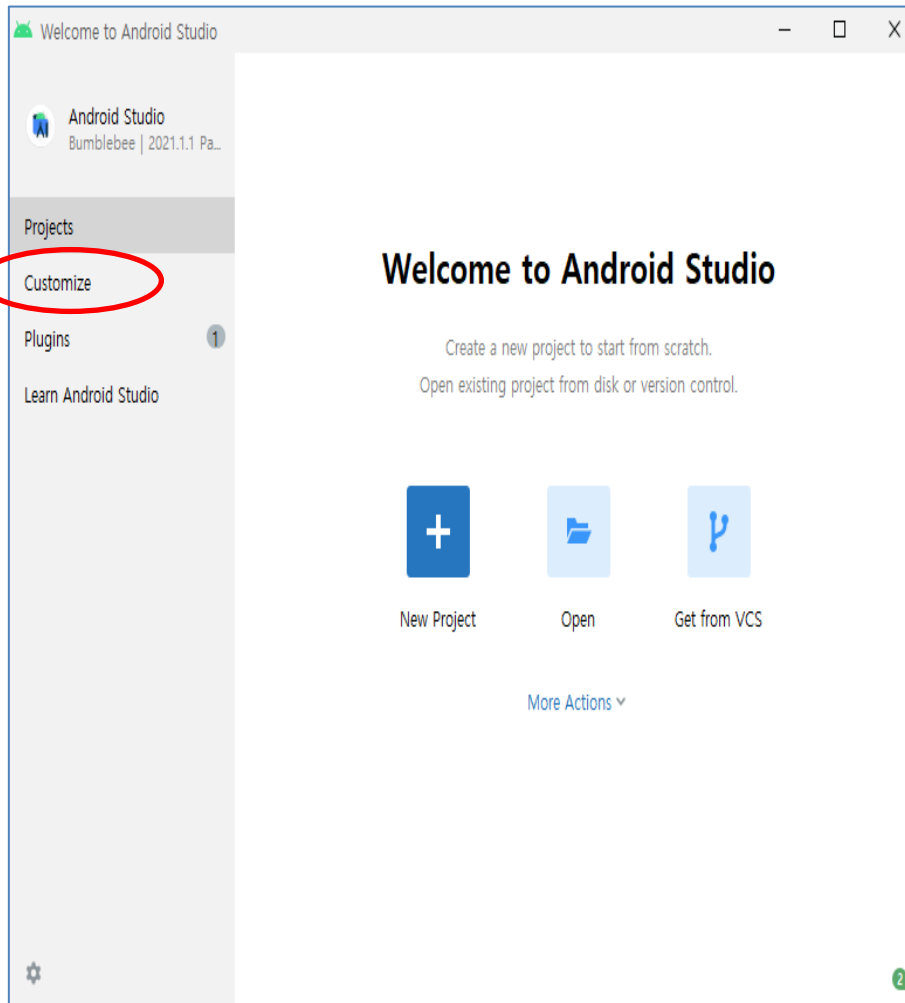
출시 노트 읽기 



본인 OS에 맞는 버전 다운로드

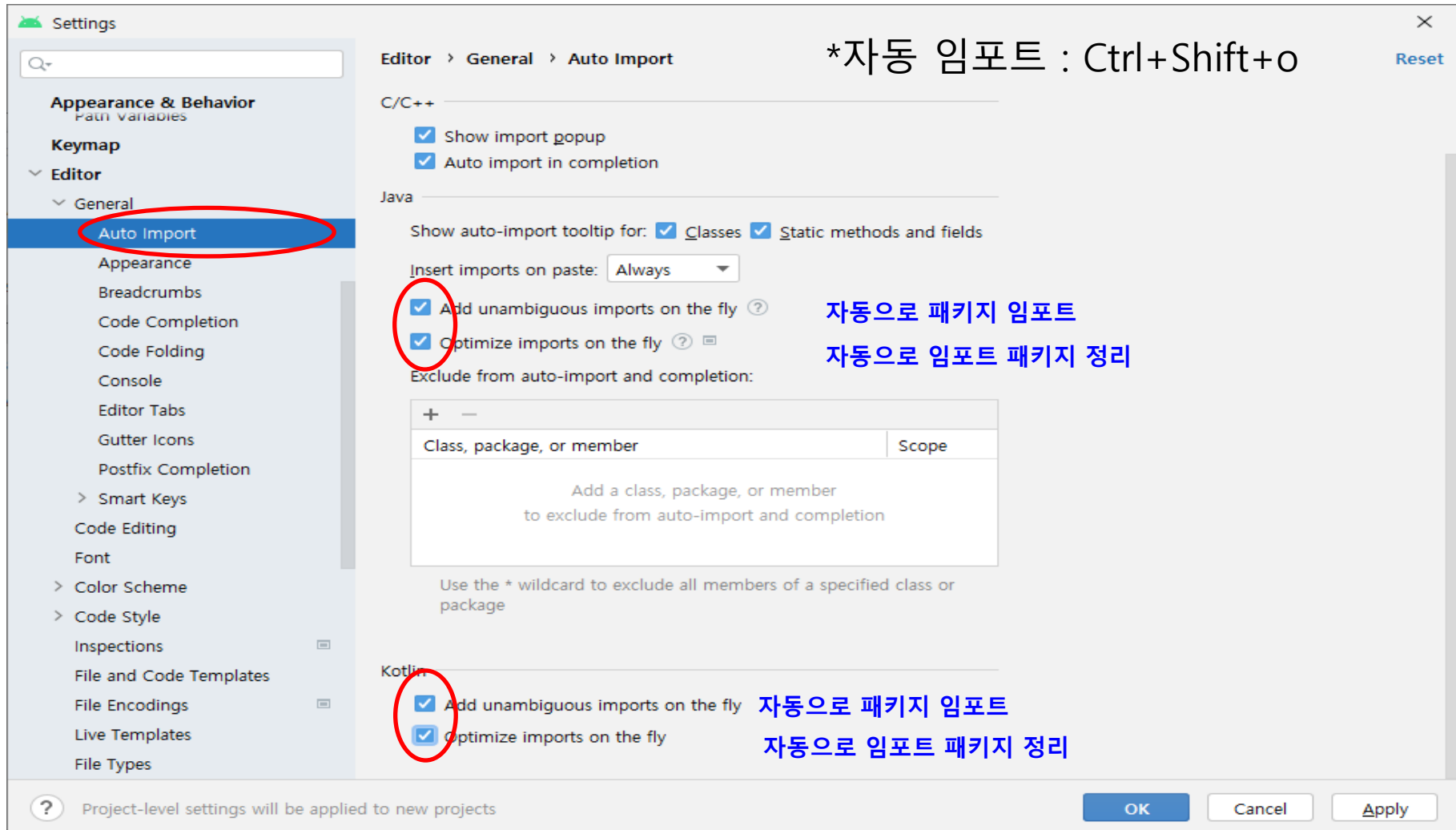
Android Studio

- 환경설정 (Setting)



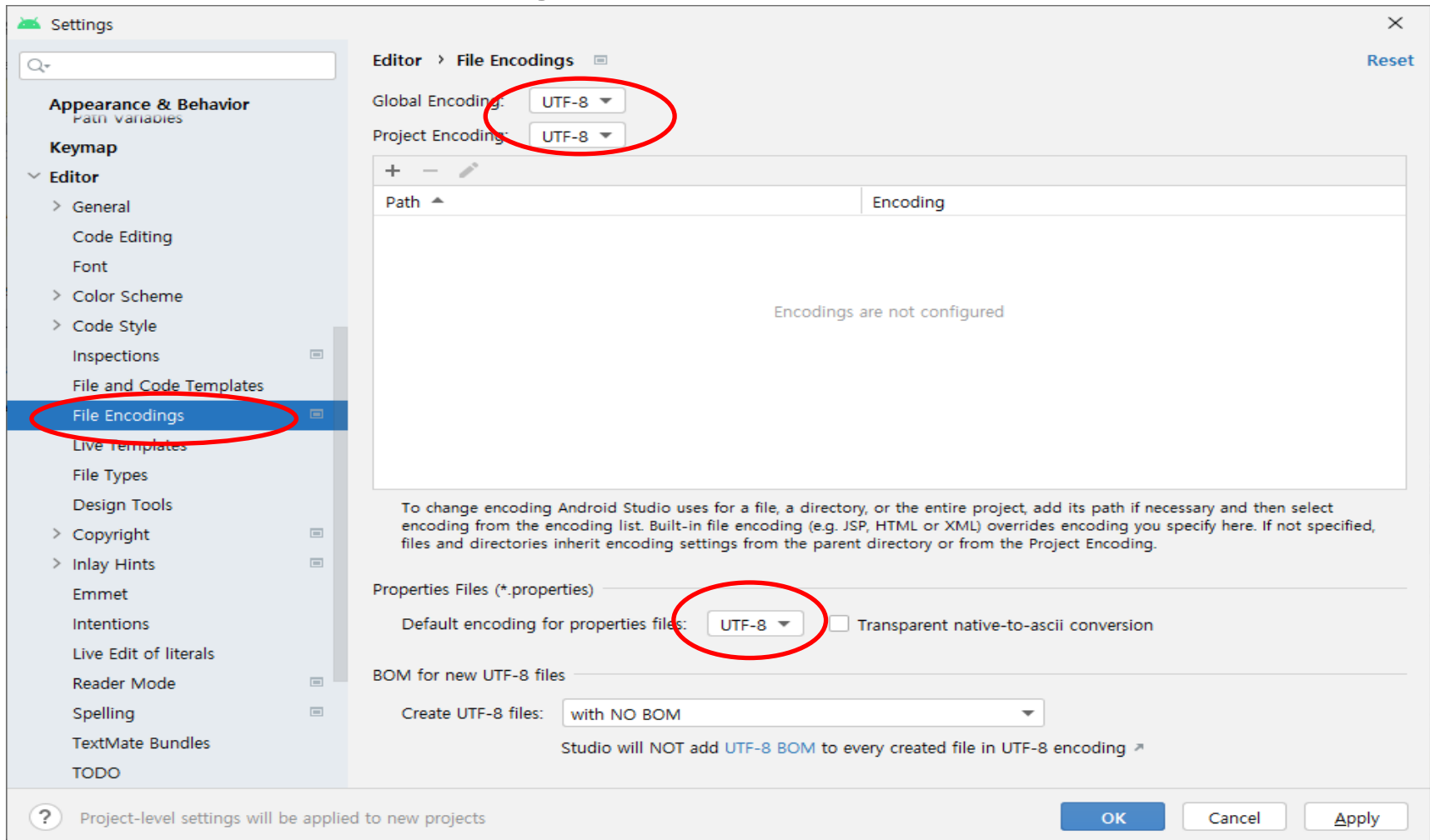
Android Studio : Settings

- Editor > General > Auto import

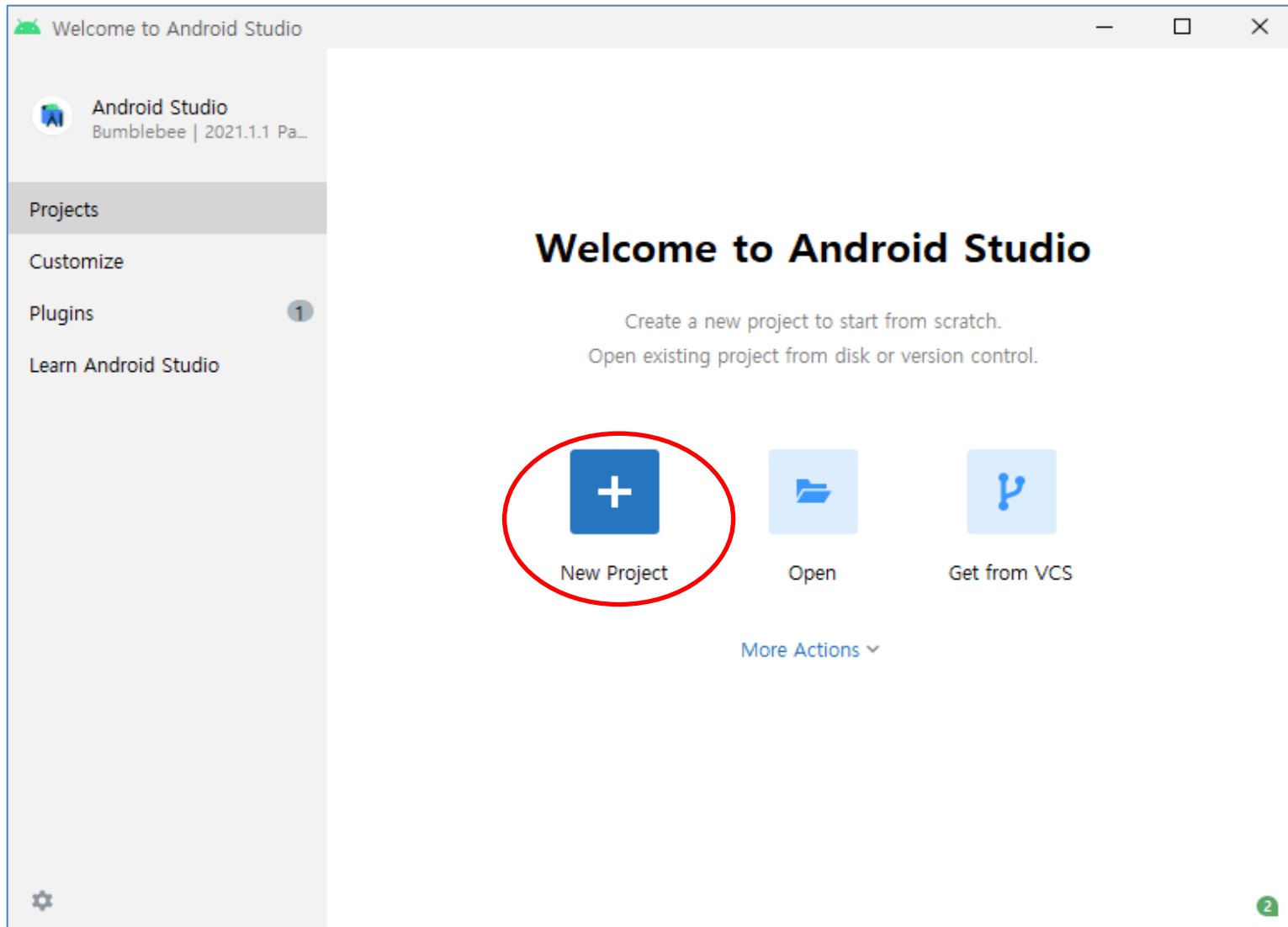


Android Studio : Settings

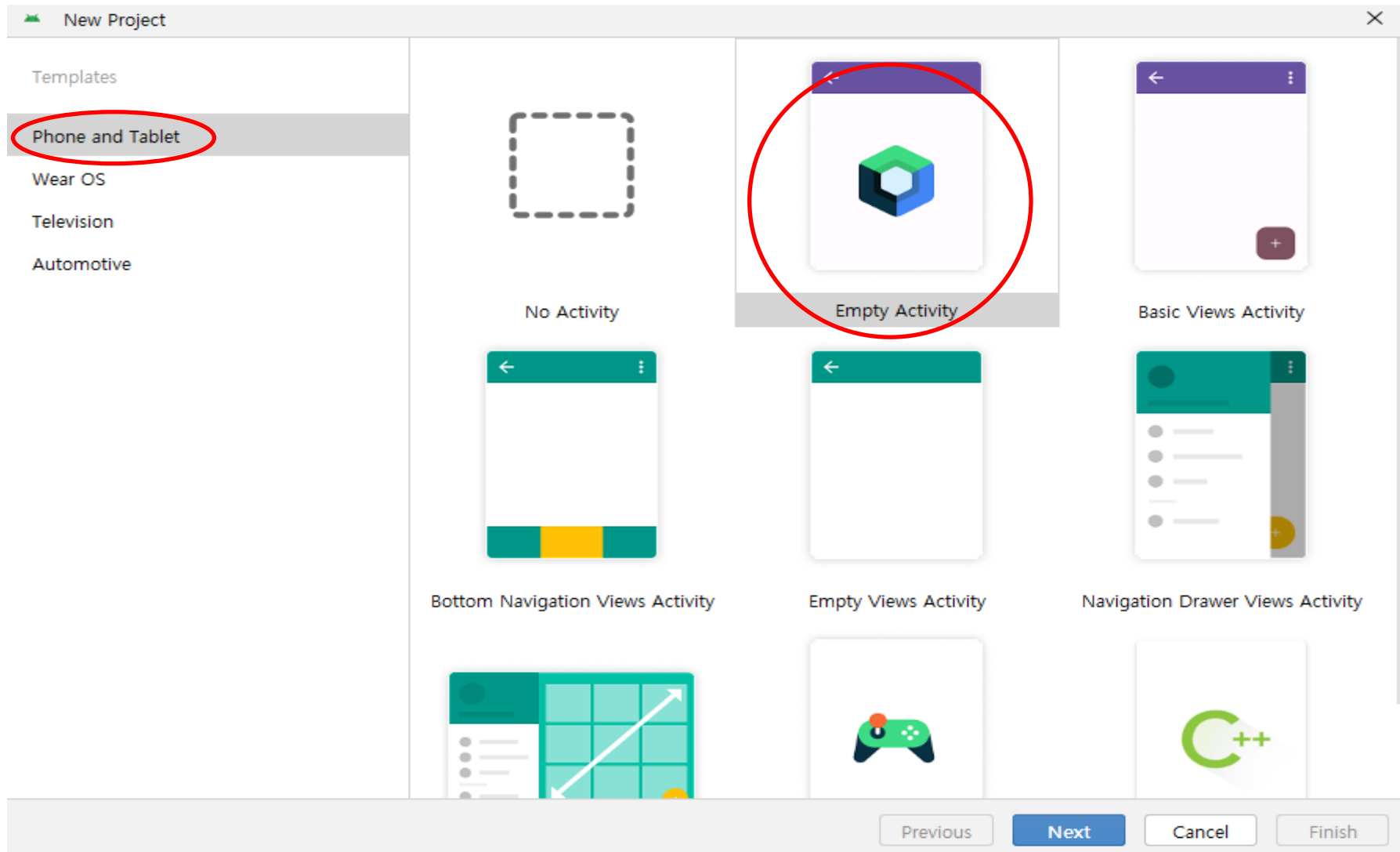
- Editor > File Encodings



프로젝트 생성



프로젝트 생성



프로젝트 생성

The screenshot shows the 'New Project' dialog in Android Studio. The dialog has a title bar 'New Project' and a close button. The main content area is divided into sections for project configuration. Annotations include a speech bubble pointing to the 'Name' field with the text '앱 이름' (App Name), a box pointing to the 'Package name' field with the text '구글 플레이에 앱을 공개할 때 사용할 고유 ID' (Unique ID used when publishing the app to Google Play), a box pointing to the 'Save location' field with the text '프로젝트 생성되는 경로를 수정할 수 있음' (The path where the project is created can be modified), and a box pointing to the 'Language' field with the text '1) 직접 수정 가능' (1) Direct modification possible and '2) 버튼으로 찾아서 가능' (2) Possible by finding with a button. The 'Finish' button is highlighted in blue.

New Project

Empty Views Activity
Creates a new empty activity

Name

Package name

Save location

Language

Minimum SDK

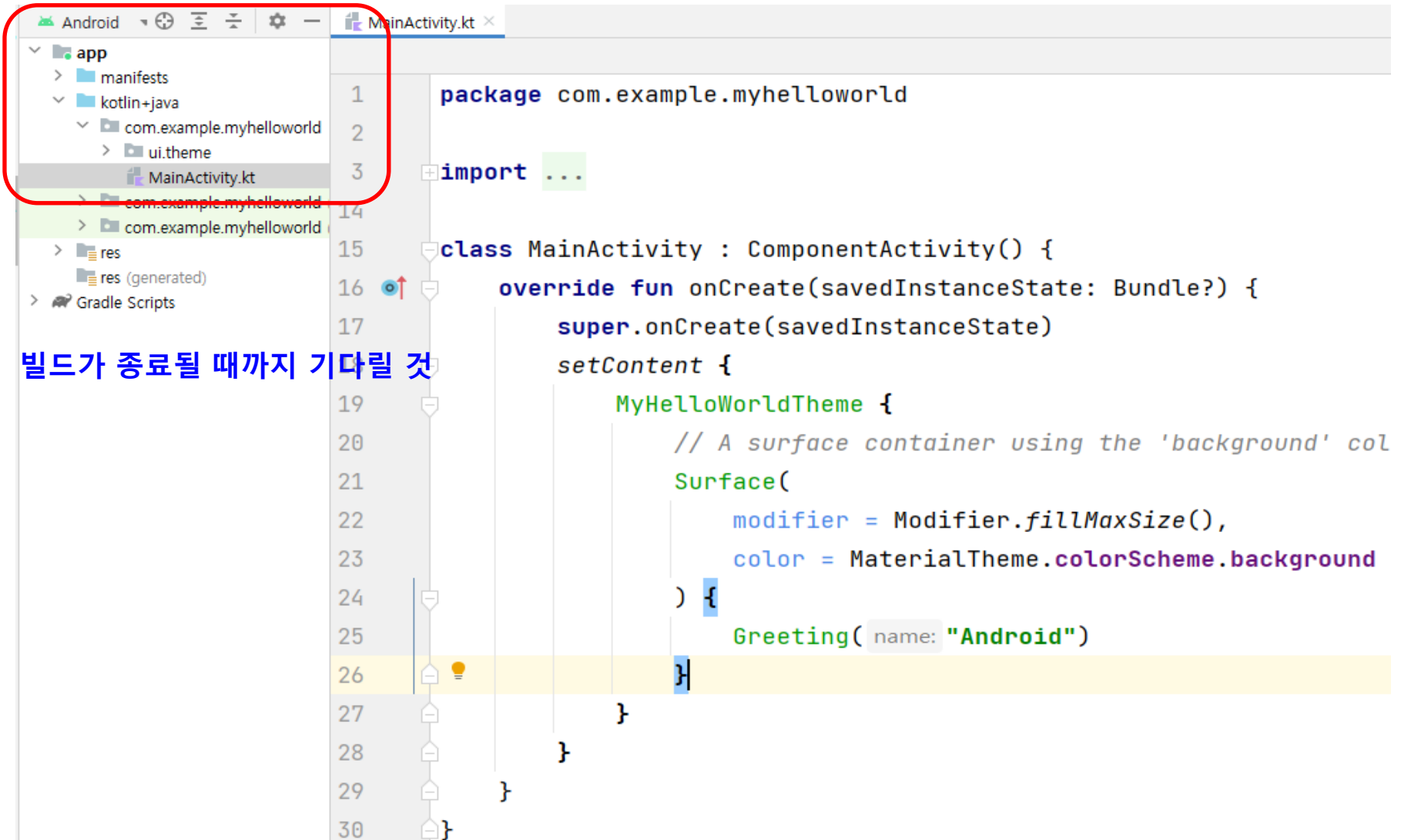
Build configuration language

Annotations:

- 앱 이름 (App Name)
- 구글 플레이에 앱을 공개할 때 사용할 고유 ID (Unique ID used when publishing the app to Google Play)
- 프로젝트 생성되는 경로를 수정할 수 있음 (The path where the project is created can be modified)
 - 1) 직접 수정 가능 (1) Direct modification possible)
 - 2) 버튼으로 찾아서 가능 (2) Possible by finding with a button)

Buttons: Previous, Next, Cancel, Finish

프로젝트 생성



Android MainActivity.kt

```
1 package com.example.myhelloworld
2
3 import ...
4
14
15 class MainActivity : AppCompatActivity() {
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18         setContentView {
19             MyHelloWorldTheme {
20                 // A surface container using the 'background' col
21                 Surface(
22                     modifier = Modifier.fillMaxSize(),
23                     color = MaterialTheme.colorScheme.background
24                 ) {
25                     Greeting( name: "Android")
26                 }
27             }
28         }
29     }
30 }
```

빌드가 종료될 때까지 기다릴 것

SDK 설치하기

- Tools >> SDK Manager

Settings

Search

- > Appearance & Behavior
- Keymap
- > Editor
- Plugins
- > Version Control
- > Build, Execution, Deployment
- > Languages & Frameworks
 - > C/C++
 - > Schemas and DTDs
 - Android SDK**
 - > Kotlin
 - Markdown
 - Template Data Languages
- > Tools
- Advanced Settings
- Kotlin Compiler
- > Experimental

Languages & Frameworks > Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: [Edit](#) [Optimize disk space](#)

SDK Platforms SDK Tools SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.

	Name	API Level	Revision	Status
<input type="checkbox"/>	Android UpsideDownCakePrivacySandbox Preview	UpsideDownCakePrivacySandbox	3	Not installed
<input type="checkbox"/>	Android 14.0 ("UpsideDownCake")	34	2	Partially installed
<input type="checkbox"/>	Android 14.0 ("UpsideDownCake")	34-ext8	1	Not installed
<input type="checkbox"/>	Android 14.0 ("UpsideDownCake")	34-ext10	1	Not installed
<input type="checkbox"/>	Android TiramisuPrivacySandbox Preview	TiramisuPrivacySandbox	9	Not installed
<input checked="" type="checkbox"/>	Android 13.0 ("Tiramisu")	33	3	Installed
<input type="checkbox"/>	Android 13.0 ("Tiramisu")	33-ext4	1	Not installed
<input type="checkbox"/>	Android 13.0 ("Tiramisu")	33-ext5	1	Not installed
<input checked="" type="checkbox"/>	Android 12L ("Sv2")	32	1	Installed
<input checked="" type="checkbox"/>	Android 12.0 ("S")	31	1	Installed
<input checked="" type="checkbox"/>	Android 11.0 ("R")	30	3	Installed
<input checked="" type="checkbox"/>	Android 10.0 ("Q")	29	5	Installed
<input type="checkbox"/>	Android 9.0 ("Pie")	28	6	Not installed
<input type="checkbox"/>	Android 8.1 ("Oreo")	27	3	Not installed
<input type="checkbox"/>	Android 8.0 ("Oreo")	26	2	Not installed
<input type="checkbox"/>	Android 7.1.1 ("Nougat")	25	3	Not installed
<input type="checkbox"/>	Android 7.0 ("Nougat")	24	2	Not installed
<input type="checkbox"/>	Android 6.0 ("Marshmallow")	23	3	Not installed

☒ Hide Obsolete Packages ☐ Show Package Details

? OK Cancel Apply

SDK 설치하기

Settings

Appearance & Behavior

Keymap

Editor

Plugins

Version Control

Build, Execution, Deployment

Languages & Frameworks

C/C++

Schemas and DTDs

Android SDK

Kotlin

Markdown

Template Data Languages

Tools

Advanced Settings

Kotlin Compiler

Experimental

Languages & Frameworks > Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\Users\greenjoa\AppData\Local\Android\Sdk Edit Optimize disk space

SDK Platform SDK Tools SDK Update Sites

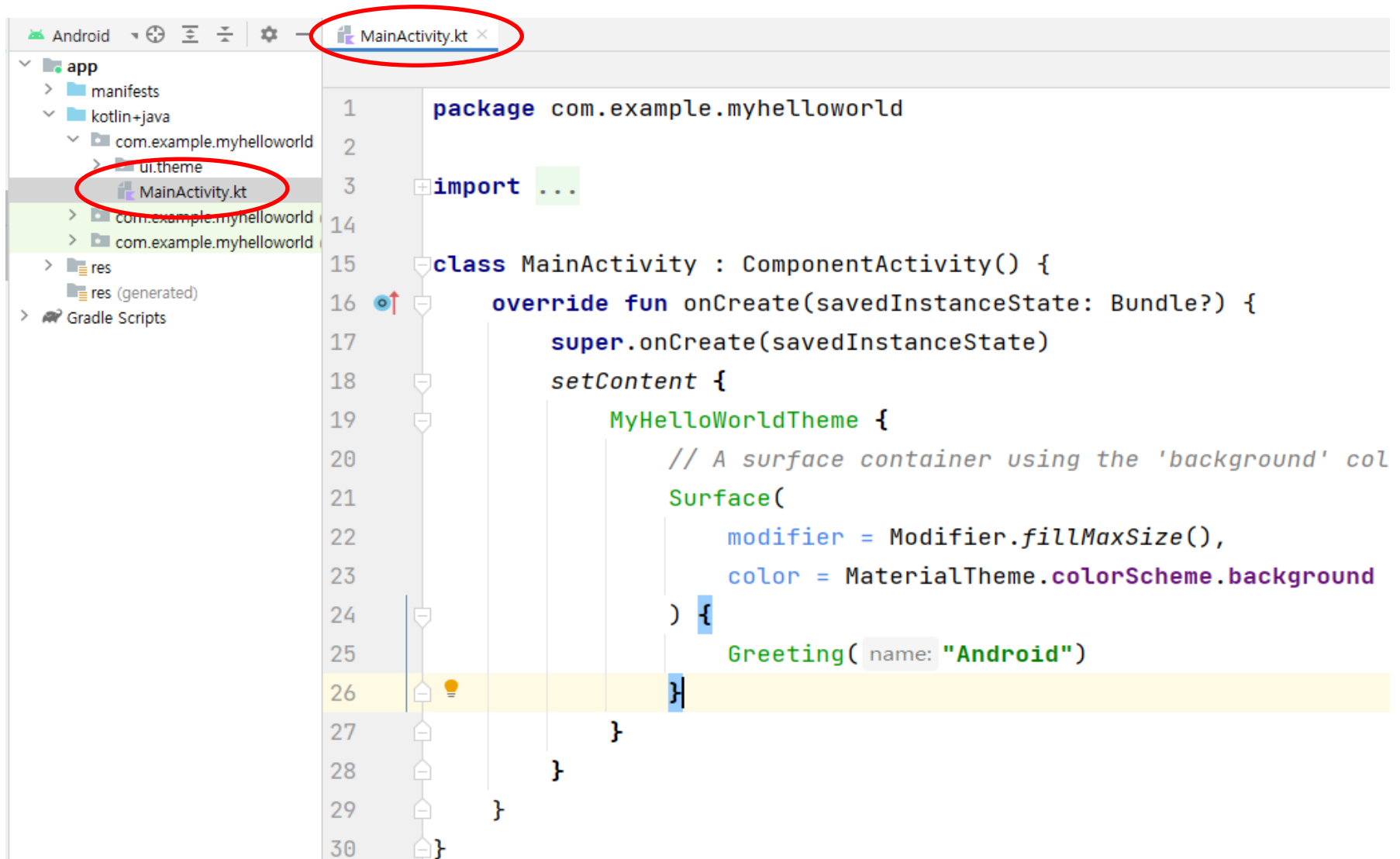
Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
<input checked="" type="checkbox"/> Android SDK Build-Tools 34		Installed
<input type="checkbox"/> NDK (Side by side)		Not Installed
<input type="checkbox"/> Android SDK Command-line Tools (latest)		Not Installed
<input type="checkbox"/> CMake		Not Installed
<input type="checkbox"/> Android Auto API Simulators	1	Not installed
<input type="checkbox"/> Android Auto Desktop Head Unit Emulator	2.0	Not installed
<input checked="" type="checkbox"/> Android Emulator	32.1.12	Update Available: 33.1.24
<input type="checkbox"/> Android Emulator hypervisor driver (installer)	2.0.0	Not installed
<input checked="" type="checkbox"/> Android SDK Platform-Tools	34.0.1	Update Available: 34.0.5
<input type="checkbox"/> Google Play APK Expansion library	1	Not installed
<input type="checkbox"/> Google Play Instant Development SDK	1.9.0	Not installed
<input type="checkbox"/> Google Play Licensing Library	1	Not installed
<input checked="" type="checkbox"/> Google Play services	49	Installed
<input type="checkbox"/> Google USB Driver	13	Not installed
<input type="checkbox"/> Google Web Driver	2	Not installed
<input type="checkbox"/> Intel x86 Emulator Accelerator (HAXM installer) - Deprecated	7.6.5	Not installed
<input type="checkbox"/> Layout Inspector image server for API 29-30	6	Not installed
<input type="checkbox"/> Layout Inspector image server for API 31-34	3	Not installed
<input type="checkbox"/> Layout Inspector image server for API S	3	Not installed

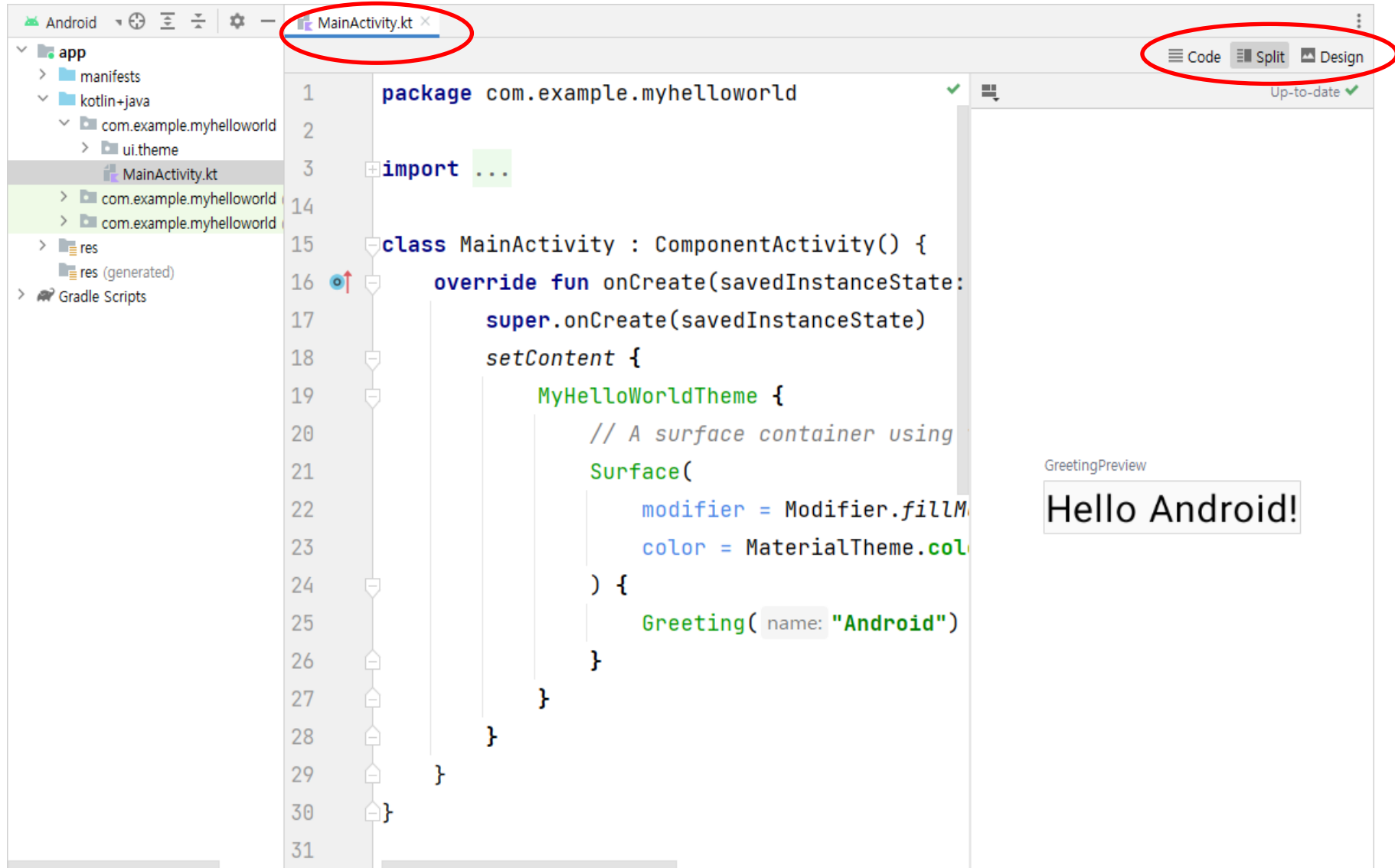
☒ Hide Obsolete Packages ☐ Show Package Details

OK Cancel Apply

애플리케이션의 구성



애플리케이션의 구성

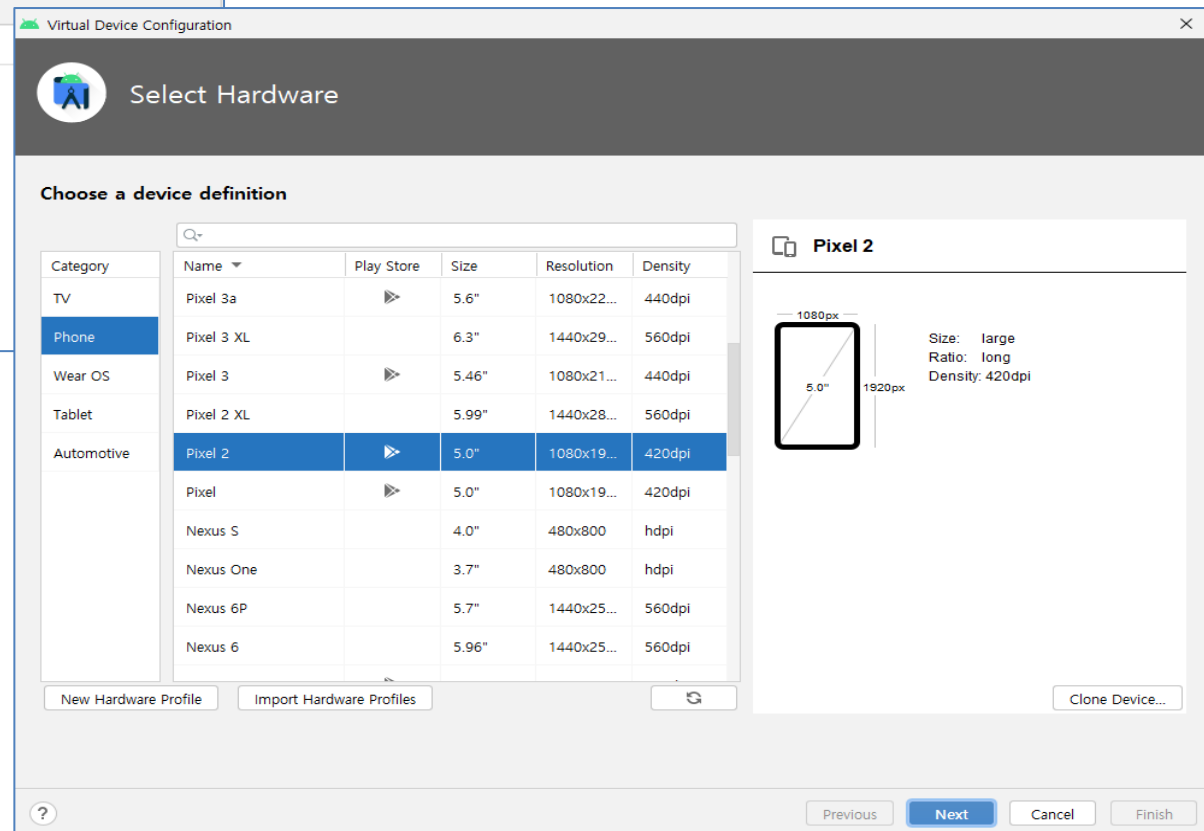
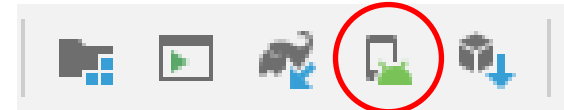
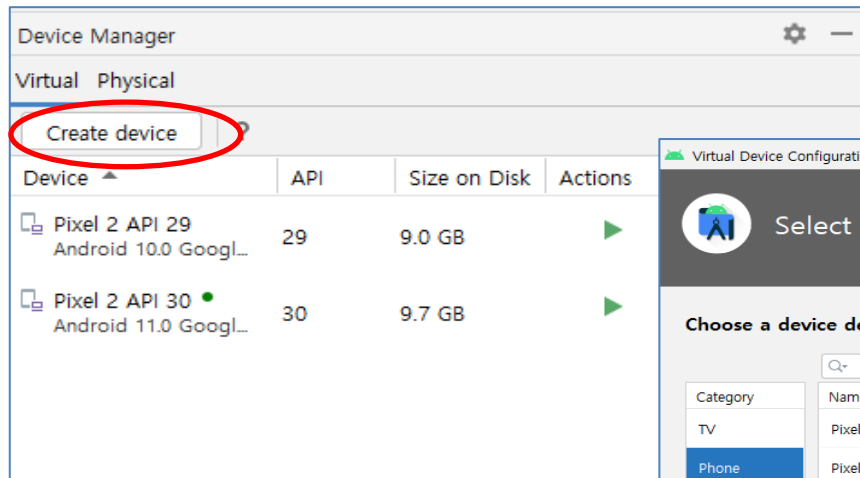


패키지 폴더

폴더 및 파일	설명
manifests	XML 파일로 애플리케이션의 전반적인 정보 즉 이름이나 내장 컴포넌트 구성과 같은 정보를 가지고 있음
java+Kotlin	소스 파일들이 들어 있는 폴더
generatedJava	Java와 관련된 각종 파일 및 라이브러리를 생성해 놓은 파일
Res	각종 리소스(자원)들이 저장되는 폴더 -. Drawable : 해상도 별도 이미지 파일들이 저장 -. Layout : 화면의 구성을 정의하는 xml 파일 -. Mipmap : 런처 아이콘 저장 -. Values : 문자열과 같은 리소스가 저장
gradle scripts	Gradle이 빌드시에 필요한 스크립트

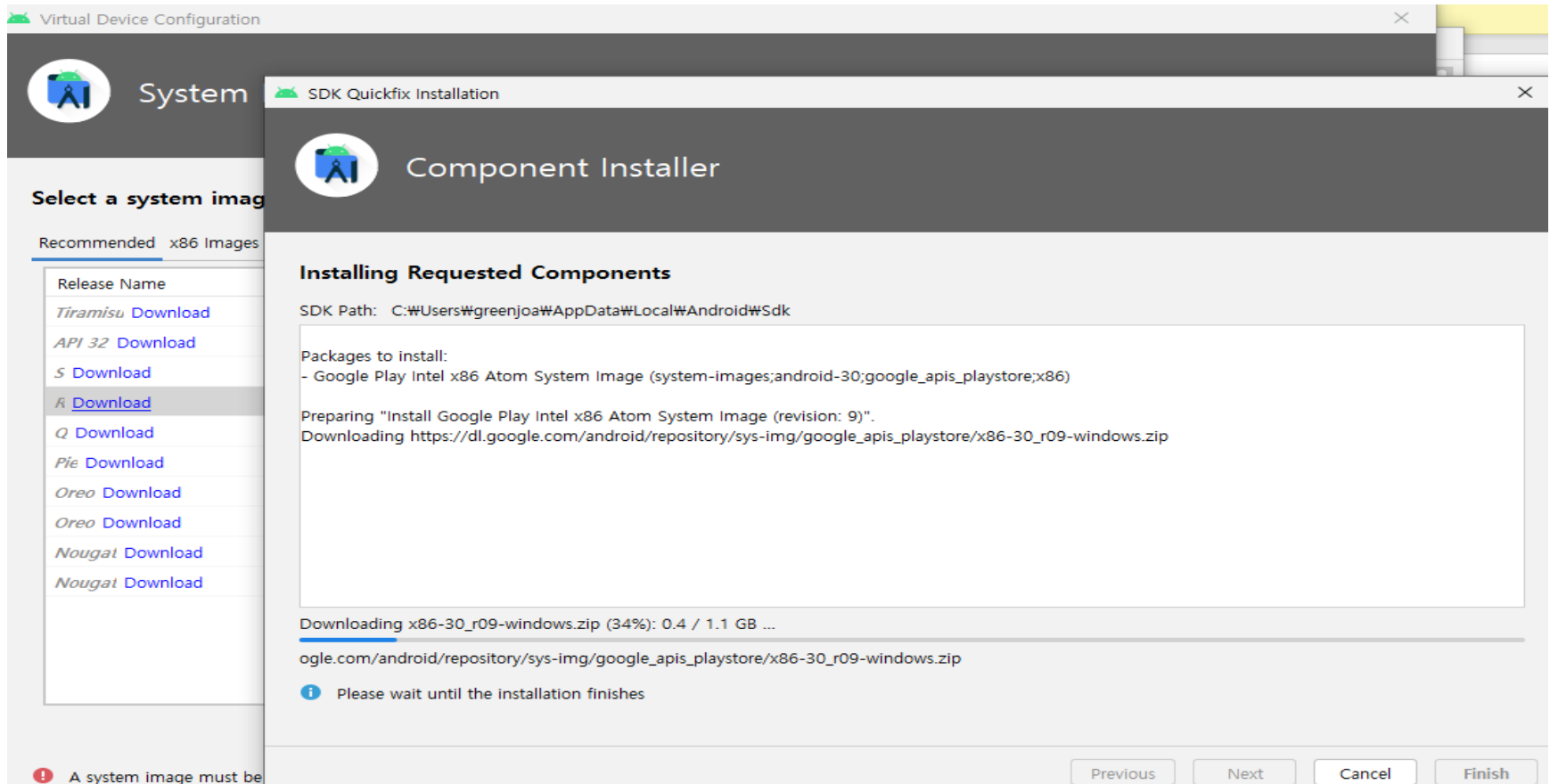
Virtual Device 생성

- Tools >> AVD Manager



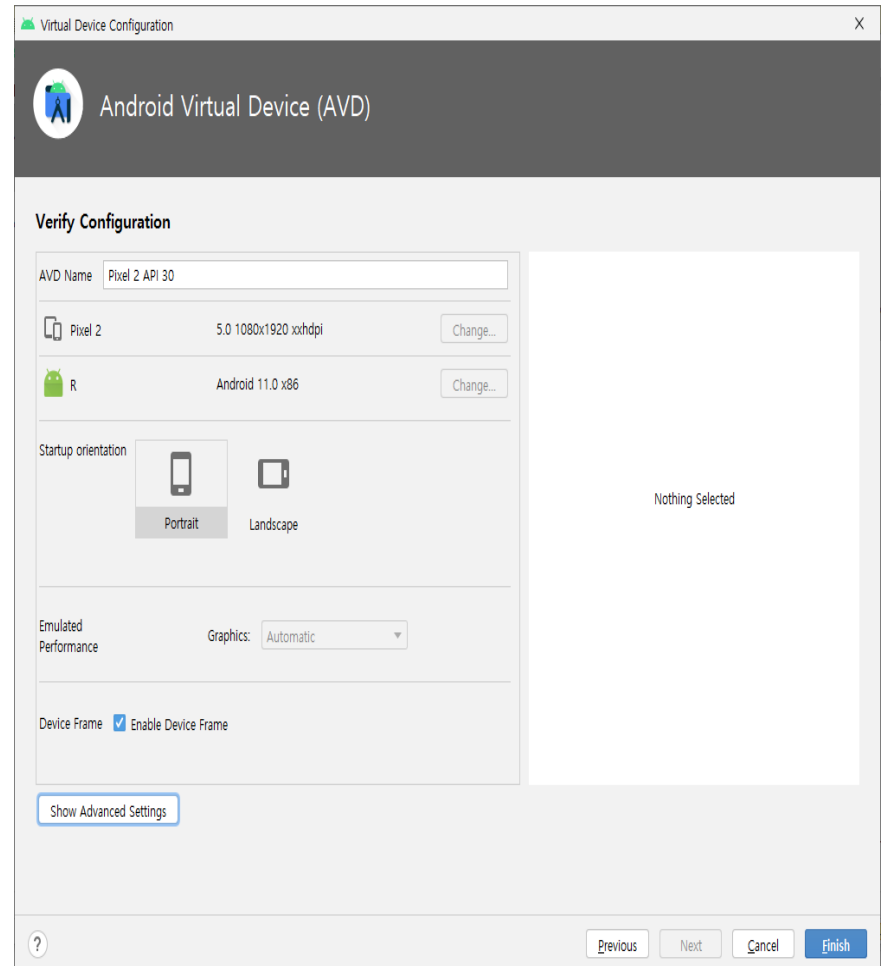
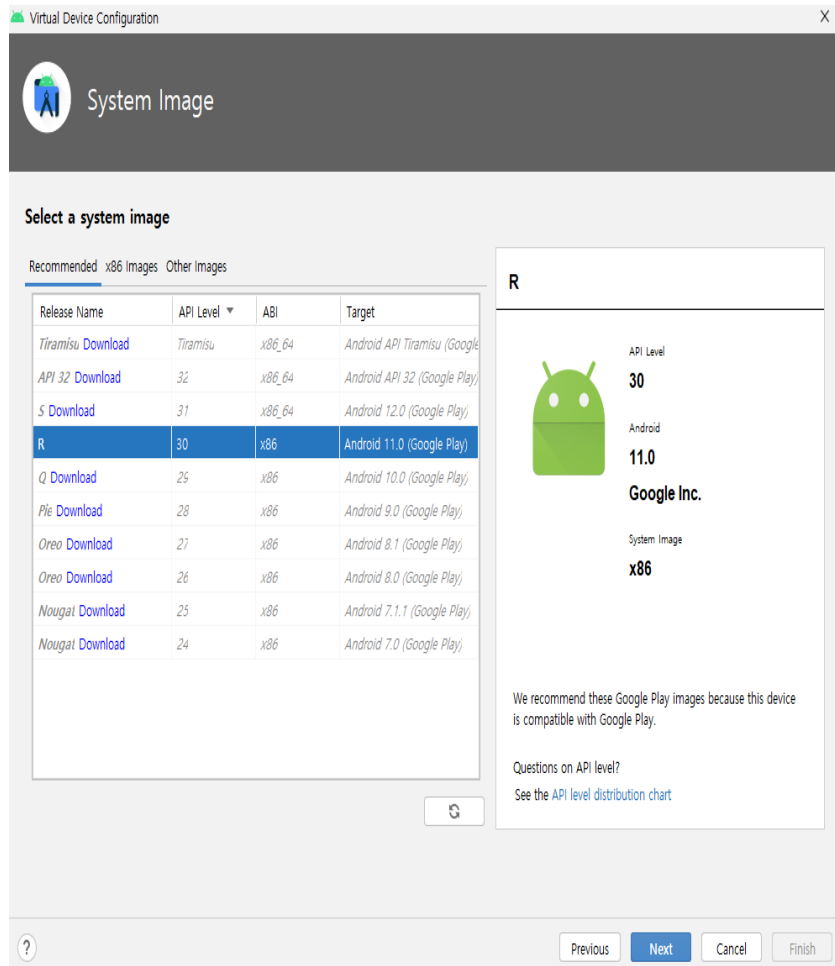
Virtual Device 생성

- System image 선택 및 다운로드



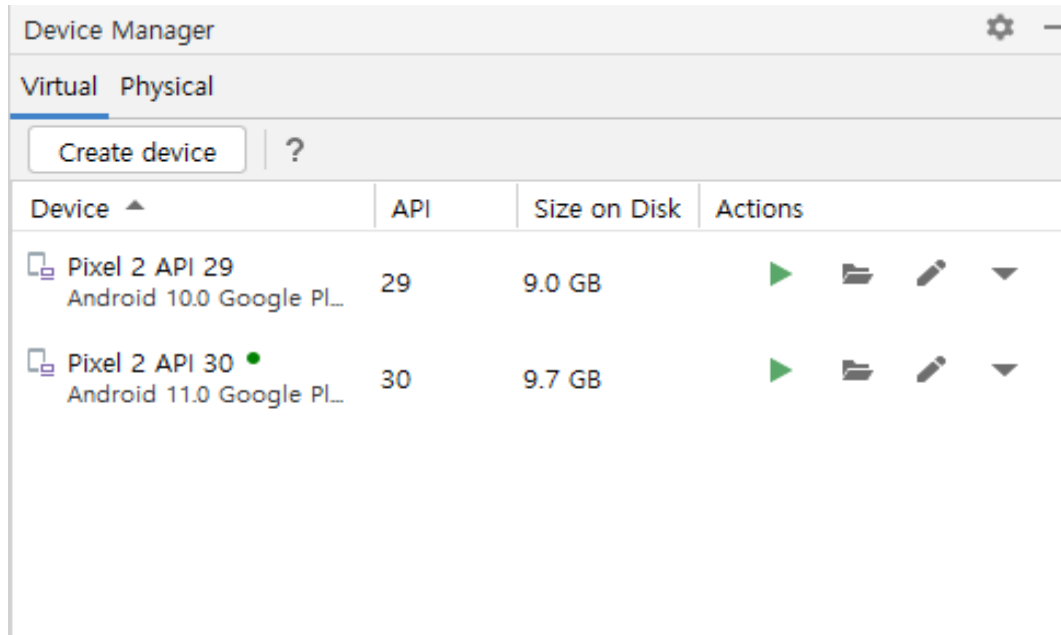
Virtual Device 생성

- Device 구성 확인 및 수정



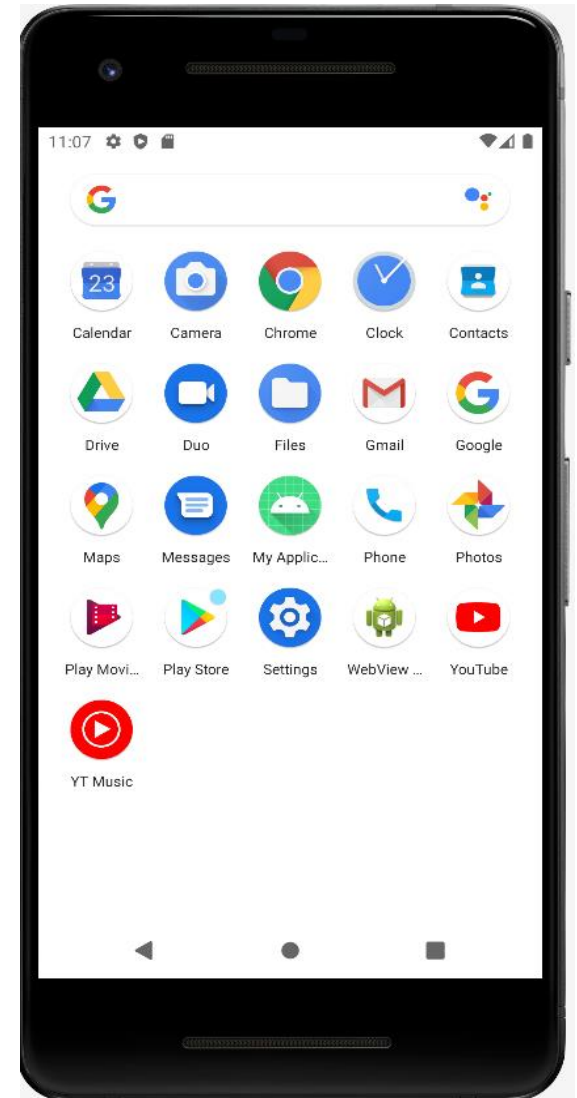
Virtual Device 생성

- 생성된 기기 확인 및 실행

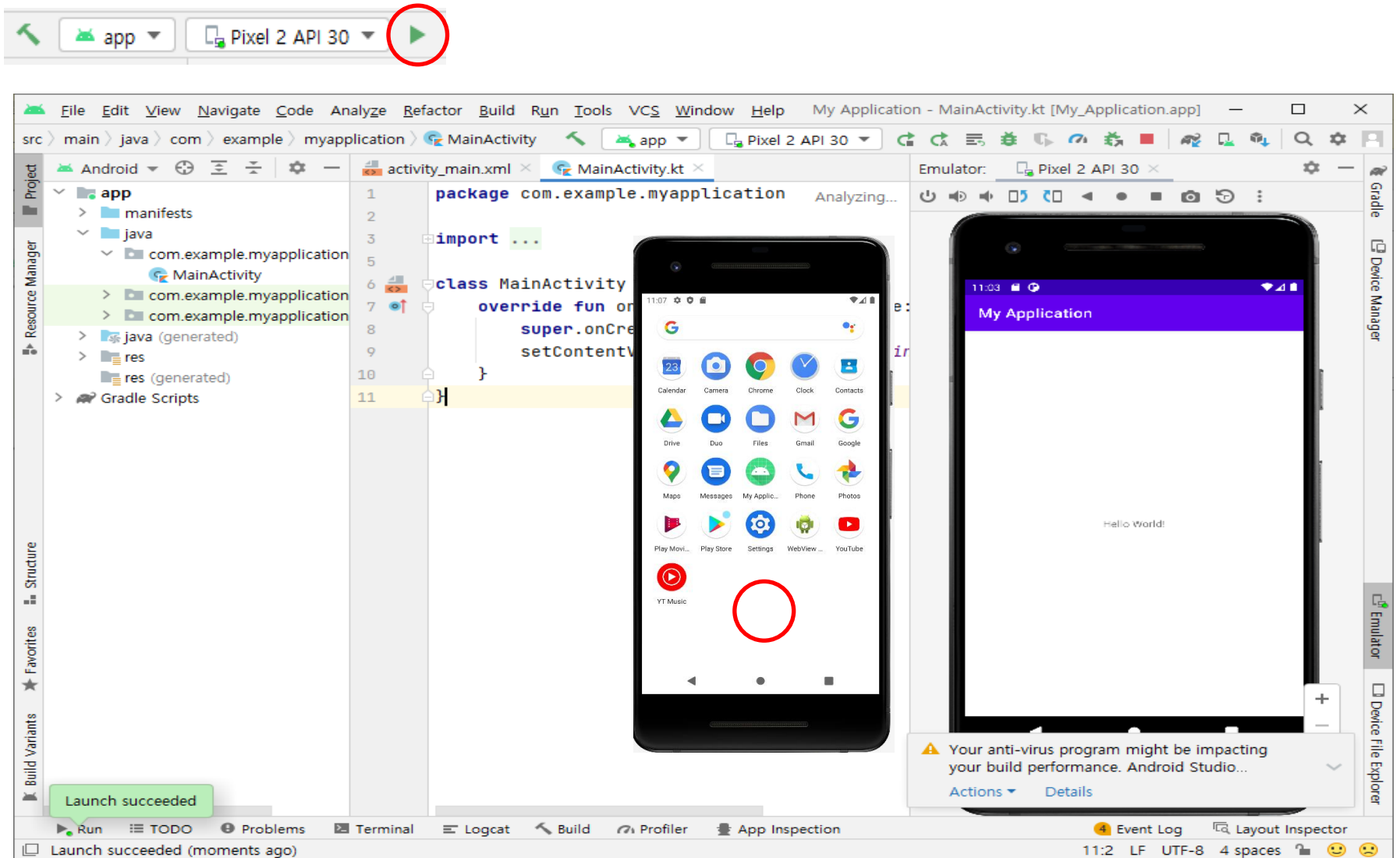


The screenshot shows the 'Device Manager' window in Android Studio. It has two tabs: 'Virtual' and 'Physical'. The 'Virtual' tab is active, showing a list of virtual devices. There are two devices listed: 'Pixel 2 API 29' (Android 10.0 Google Pixel) and 'Pixel 2 API 30' (Android 11.0 Google Pixel). The 'API' column shows 29 and 30 respectively, and the 'Size on Disk' column shows 9.0 GB and 9.7 GB. The 'Actions' column for each device contains icons for running (green play button), deleting (trash can), editing (pencil), and a dropdown menu (arrow).

Device	API	Size on Disk	Actions
Pixel 2 API 29 Android 10.0 Google PL...	29	9.0 GB	[Run] [Delete] [Edit] [Dropdown]
Pixel 2 API 30 Android 11.0 Google PL...	30	9.7 GB	[Run] [Delete] [Edit] [Dropdown]

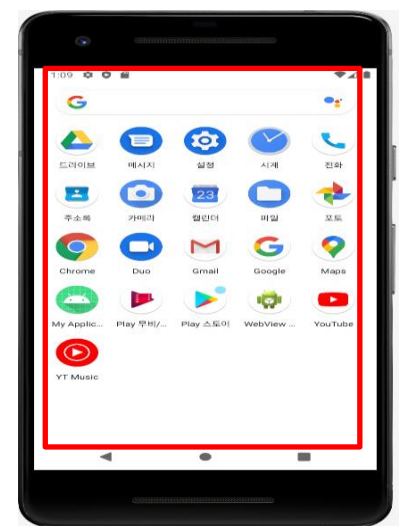
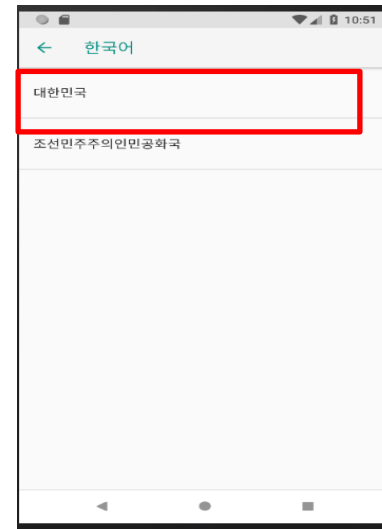
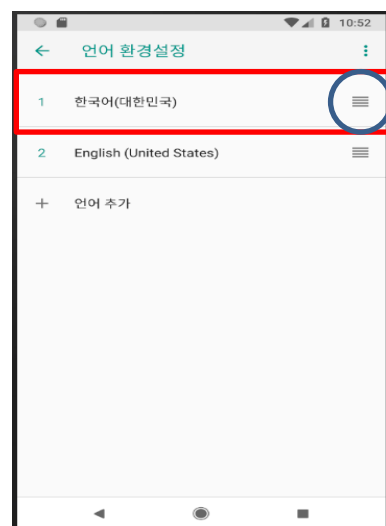
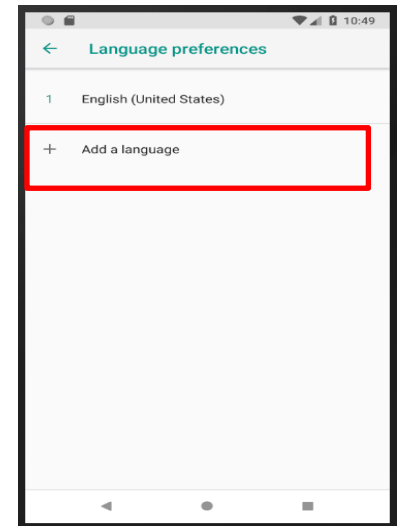
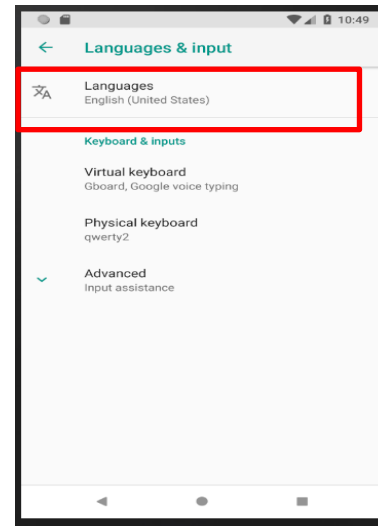
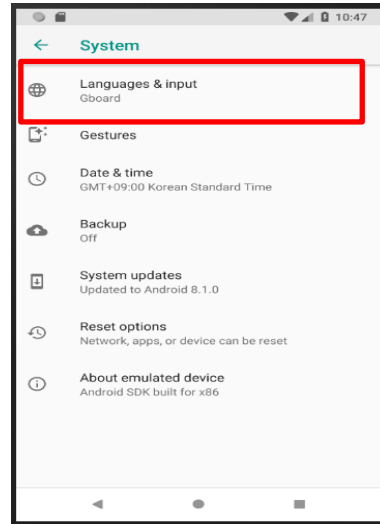
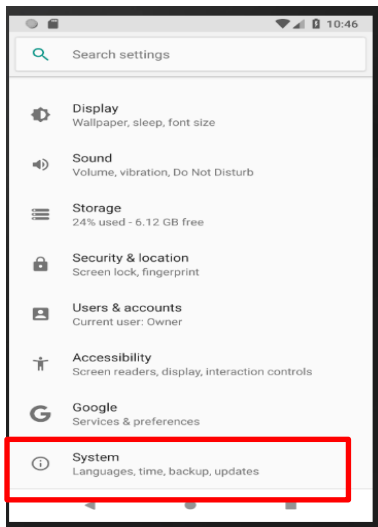


앱 설치 및 실행



한국어 설정

* Settings 앱 실행



Kotlin 소개

코틀린(Kotlin) 개요

- 코틀린(Kotlin)
 - OSS(Open Source Software)로 젯브레인(JetBrains)에서 개발한 프로그래밍 언어
 - 2011년부터 개발이 시작되어 2016년에 1.0 정식 버전이 발표
 - 2024년 2월 1.9.22
 - 구글 I/O 2017에서 코틀린을 안드로이드 애플리케이션 공식 개발언어로 발표
 - 안드로이드 애플리케이션은 자바와 코틀린으로 개발할 수 있음

<https://kotlinlang.org/>

코틀린(Kotlin)의 특징

- 코틀린 컴파일러는 JVM(자바 가상머신)에서 실행되는 자바 바이트 코드(클래스 파일)를 생성하므로, **자바와 완벽하게 호환되는 크로스 플랫폼을 지원**
 - 자바 애플리케이션이 실행 가능한 환경이면 코틀린 애플리케이션도 실행 가능
 - 다양한 애플리케이션 작성 가능
 - Backend app
 - Cross-platform mobile app
 - Android app
 - Multiplatform library
- 자바처럼 정적 타입의 언어
 - 모든 표현식의 타입은 컴파일 시점에 알 수 있음
 - 타입 추론(type inference)기능
 - **변수를 선언하고, 타입을 명시적으로 지정하지 않아도 처리해 줌**
 - 예) `val a= 10 // val a : Int = 10`

코틀린(Kotlin)의 특징

- 자바와 동일하게 객체지향 프로그래밍을 지원함
- 함수형 프로그래밍을 지원 함
 - 클래스와는 별도로 함수를 선언할 수 있음
 - 다양한 형태의 함수와 람다식을 지원함
- 문법과 코드가 간결하여 작성 부담이 줄어 듦
 - 클래스의 getter와 setter 등이 자동 생성됨
 - 명령문 끝에 세미콜론(;)을 붙이지 않아도 됨
 - 세미콜론은 같은 줄에 여러 문장 기술할 때만 이용
- Null 값으로 인해 발생할 수 있는 NullPointerException 예외가 생기지 않도록 언어 자체에서 방지하므로, 자바보다 안전 (`var s: String? = null`)
- OSS 이므로 코틀린 및 OSS IDE(Android Studio, IntelliJ IDEA, Eclipse)도 무상으로 사용할 수 있음
 - 자바코드를 코틀린 코드로 변환하는 변환기 지원

코틀린(Kotlin) 개발 IDE



USE
IntelliJ IDEA

Bundled with Community
Edition or IntelliJ IDEA Ultimate

Instructions



USE
Android Studio

Bundled with [Studio 3.0](#), plugin
available for earlier versions

Instructions



USE
Eclipse

Install the plugin from the
Eclipse Marketplace

Instructions



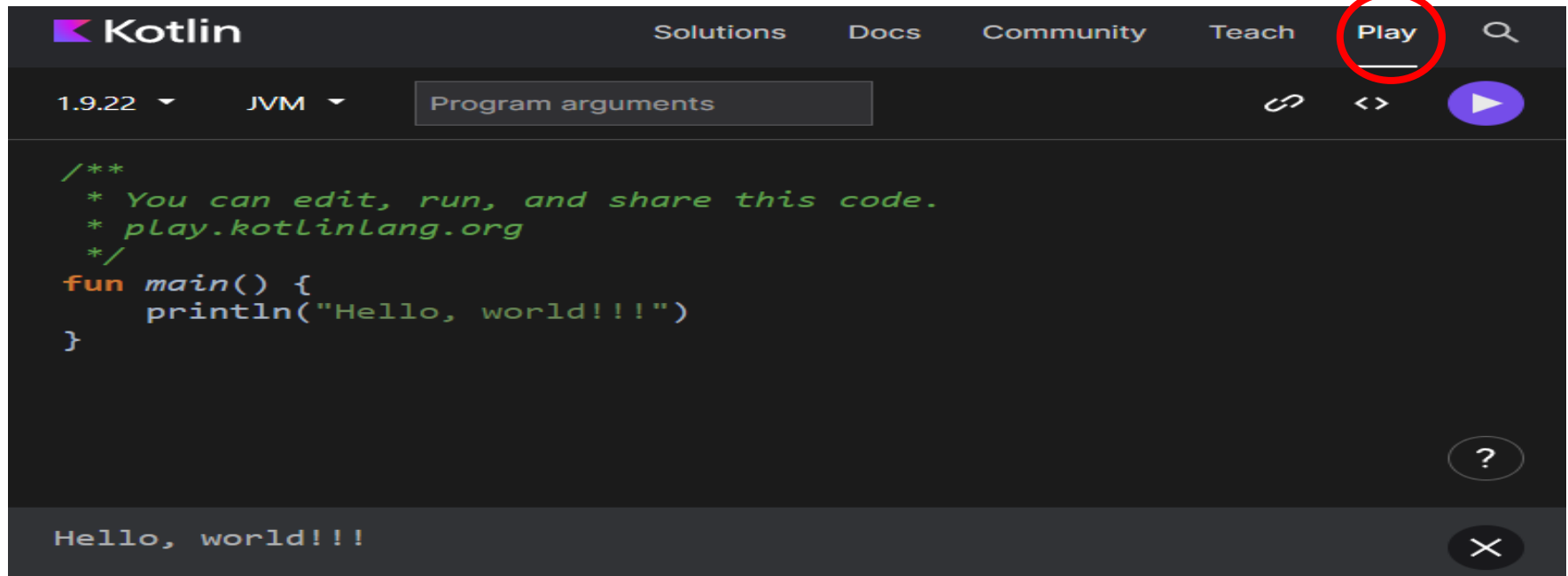
STANDALONE
Compiler

Use any editor and build from
the command line

Download Compiler

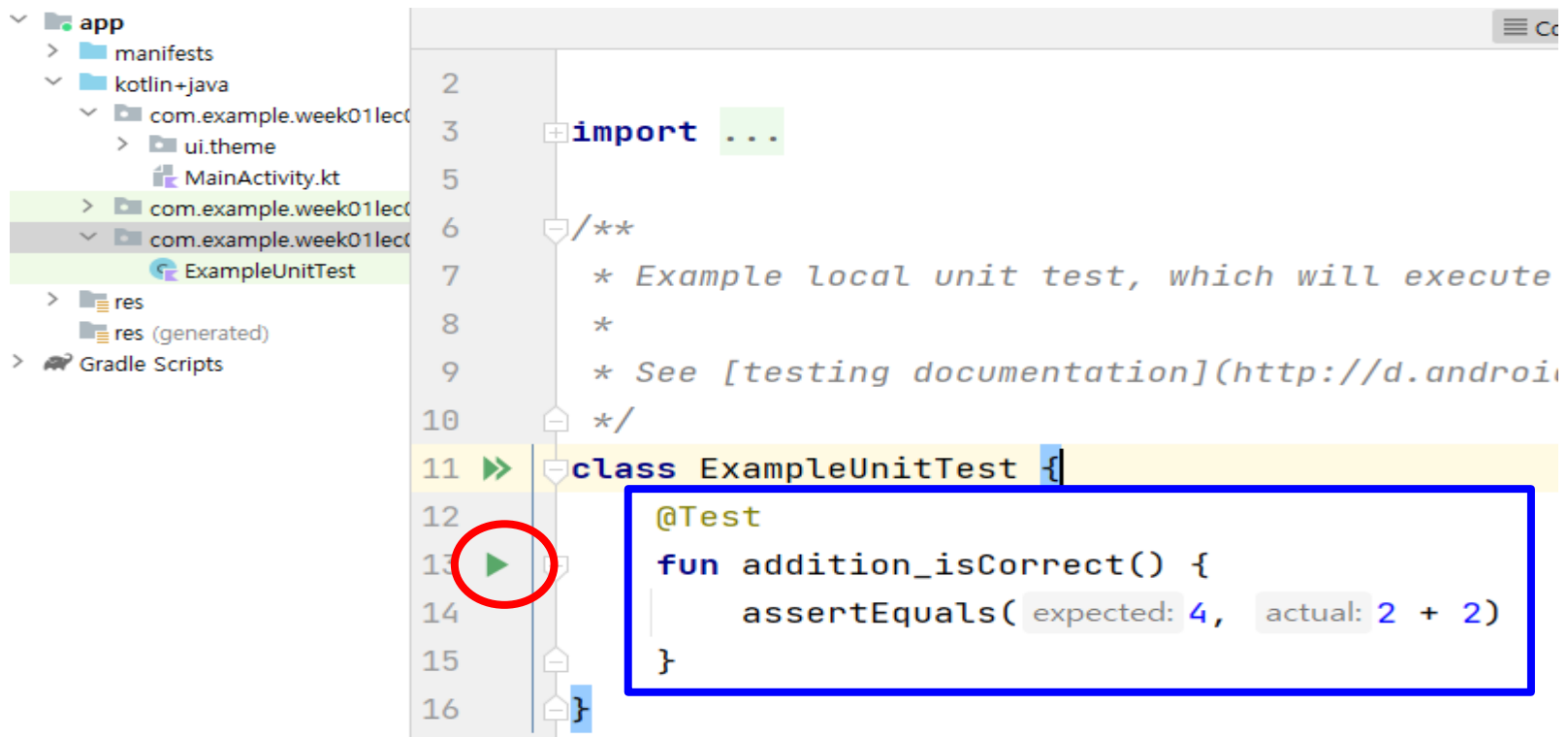
코틀린(Kotlin) 개발

- <https://kotlinlang.org/>



Android Studio에서 코틀린 사용하기

- Kotlin File/Class 추가 하여 사용
- Local UnitTest에 Kotlin File/Class 추가
 - @Test 라는 annotation을 가진 함수 추가하여 사용



데이터 타입 및 연산자

기본 타입 (Basic types)

- 코틀린에서는 모든 것이 객체
 - 멤버 함수나 프로퍼티(멤버 변수, 상수)를 호출할 수 있음
- 기본 타입
 - **Numbers : Byte, Short, Int, Long, Float, Double**
 - **Character : Char**
 - **Boolean : Boolean**
 - Array
 - String

기본 타입 : Primitive data types

Kotlin Type	Bit width	
Double	64	12.3
Float	32	12.3F / 12.3f
Long	64	123L
Int	32	123
Short	16	[-32768, 32767]
Byte	8	[-128, 127]
Boolean	1	true, false
Char	2	하나의 문자 'a'

- * 자바와 비슷하게 처리하지만, 똑같지는 않음
 - Char가 Kotlin에서는 숫자형이 아님

기본타입 : Explicit Conversions

- 주의) 작은 타입은 큰 타입의 하위 타입이 아님

- 작은 타입에서 큰 타입으로 대입이 안됨

```
val a: Int = 1
```

```
val b: Long = a // 오
```

- 명시적 변환

```
val a: Int = 1
```

```
val b: Long = a.toLong()
```

< 타입 변환 함수들 >

- toByte(): Byte
- toShort(): Short
- toInt(): Int
- toLong(): Long
- toFloat(): Float
- toDouble(): Double
- toChar(): Char

기본 타입 : Implicit conversions

- 산술 연산에서는 묵시적 변환이 이루어짐

```
val num = 1L + 3 // Long + Int => Long
```

- Operations
 - Kotlin은 numbers에 대해 표준 산술 연산들을 지원함
 - Bitwise operations

val x = (1 shl 2) and 0x000FF000

- shl(bits) – signed shift left (Java's <<)
- shr(bits) – signed shift right (Java's >>)
- ushr(bits) – unsigned shift right (Java's >>>)
- and(bits) – bitwise and
- or(bits) – bitwise or
- xor(bits) – bitwise xor
- inv() – bitwise inversion

기본 타입 : Characters

- Char는 숫자로 취급되지 않음

```
fun check(c: Char) {  
    if (c == 1) { // ERROR: incompatible types  
        // ...  
    }  
}
```

```
fun check(c: Char) {  
    if (c == 'a') {  
        // ...  
    }  
}
```

```
fun decimalDigitValue(c: Char): Int {  
    if (c !in '0'..'9')  
        throw IllegalArgumentException("Out of  
range")  
    return c.code - '0'.code  
}
```

*code 는 toInt()의 기능을 수행하는 함수

변수 및 상수

변수 및 상수

- 클래스의 멤버 변수 및 전역 변수는 반드시 선언하면서 초기화해야 함
- 지역 변수의 경우, 선언 후 사용 가능
 - 단, 초기화가 되어 있지 않으면 사용할 때 에러 발생함

변수 및 상수

- 변수 (Variable, mutable variable)

Kotlin

var variable:Type = value

var a: Int = 7

var b: Double = 10.5

var c: String = "greenjoa"

Java

Type variable = value;

int a = 7;

double b = 10.5;

String c = "greenjoa";

변수 및 상수

- 상수 (value, immutable variable)

Kotlin

val variable:Type = value

val a: Int = 7

val b: Double = 10.5

val c: String = "greenjoa"

Java

final Type variable = value;

final int a = 7;

final double b = 10.5;

final String c = "greenjoa";

변수 및 상수

- 숫자 타입

```
val a=100          // 자동 추론 (Int)
```

```
val b: Long = a    // 숫자 타입간 변환 자동으로 해주지 않음
```

```
val b: Long = a.toLong() // 변환 함수 사용
```

```
val c: Long = a + 1L   // 표현식을 Long 타입으로 변환
```


변수 및 상수

- 문자열 : String 타입

```
val s1 = "홍길동"
```

```
val s2 = "5".toInt()
```

```
val s3 = "123.5".toDouble()
```

- 문자열 내에서 변수나 상수 출력

“ `${표현식}` ”

```
println("이름: $s1 \n번호 : ${s2}번 \n응모가 : ${s3-10}원")
```

변수 및 상수

- 문자열 비교
 - == : java의 equals()와 동일

```
val str = "hello"  
if(str == "hello") {  
    println("Hi")  
} else{  
    println("안녕")  
}
```

변수 및 상수

- 코틀린에서 모든 변수는 **null을 허용하지 않음**

- 클래스의 멤버 변수 및 상수의 경우 선언과 동시에 초기화해야 함

- 기타 변수들도 null 값 저장을 허용하지 않음

- **Null을 허용하려면 물음표(?)를 추가 해야 함**

- 타입을 생략할 수 없음



String
String?
↓
다른 타입

```
var str : String? = "greenjoa"  
str = null
```

```
val a:String //오류 (멤버변수)
```

```
val b: String? = null
```

```
var c:String //오류 (멤버변수)
```

```
var d:String?= null
```

변수 및 상수 : Variable Nullable

- **lateinit** 키워드로 나중에 초기화 가능 (클래스 멤버)
 - **Var** 변수에만 사용 가능
 - Null 값으로 초기화할 수 없음
 - 초기화 전에는 사용할 수 없음
 - Int, Long, Double, Float 등 기본 데이터 타입에는 사용할 수 없음

```
lateinit var c:String
```

```
c= "Hello"
```

```
println(c)
```

변수 및 상수 : Variable Nullable

- Lazy로 늦은 초기화 (클래스 멤버)
 - Val 에 사용하여 늦은 초기화
 - 초기화 블록에 초기화에 필요한 코드 작성
 - 처음 호출될 때 초기화 블록의 코드가 실행됨
 - 마지막 줄에는 초기화할 값을 명시함

```
val str:String by lazy{  
    print("초기화")  
    "Hello"  
}
```

```
println(str) // 초기화Hello  
println(str) // Hello
```

변수 및 상수 : Variable Nullable

- Not-null assertion (!!)
 - Null이 아닌 형식에 할당하 수 있도록 함
 - Null값이 아님을 보증(!!)
 - 변수 뒤에 !!를 추가하면 null값이 아님을 보증 ➔ null일 경우 예외

```
val name:String?= "greenjoa"
```

```
val name2:String = name // 예러
```

```
val name3:String? = name // OK
```

```
val name4:String = name !! // OK
```

변수 및 상수 : Variable Nullable

- 안전한 호출 연산자(?.)
 - Null값이 아닌 경우에만 호출

AF?문프인X

val str:String?=**null**

var upperCase = **if**(str!=**null**) str **else null** // null

upperCase.toUpperCase() // 에러

upperCase = str?.toUpperCase() // null

변수 및 상수 : Variable Nullable

- 엘비스(Elvis) 연산자 (?:)
 - Null이 아닌 기본 값을 반환하고 싶을 때 사용

```
val str:String?=null
```

```
var upperCase = str?.toUpperCase() ?: "초기화하시오 "
```

```
var upperCase = str?: "초기화하시오"
```


데이터 타입 : 배열

- Array 클래스
 - Array<String>과 같이 제너릭 타입으로 나타냄
 - 배열은 크기를 변경할 수 없음
- 배열 생성 – **arrayOf** 함수

```
val item: Array<Any> = arrayOf(1, "바나나", false)
```

```
val item: Array<String> = arrayOf("사과", "바나나", "키위")
```

*** 빈 배열 생성**

```
val arr = arrayOfNulls<Int>(10)
```

데이터 타입 : 배열

- 배열 생성 – Array 생성자

```
val item2 = Array<String>(5) {i->(i*i).toString()}
```

```
item2[0] = item[1]
```

```
for(fruit in item2){
```

```
    println(fruit)
```

```
}
```

<https://github.com/JetBrains/kotlin/blob/master/core/builtins/native/kotlin/Array.kt>

```
public class Array<T> {  
    /**  
     * Creates a new array with the specified [size], where each element is calculated by calling the specified  
     * [init] function.  
     *  
     * The function [init] is called for each array element sequentially starting from the first one.  
     * It should return the value for an array element given its index.  
     */  
    public inline constructor(size: Int, init: (Int) -> T)
```

데이터 타입 : 배열

- 기본 타입 요소를 저장하는 배열 클래스

```
val item : IntArray = intArrayOf(1,2,3,4,5)
```

```
val item2 = IntArray(5) {j->(j*j)}
```

```
for(num in item2){
```

```
    println(num)
```

```
}
```

- ByteArray, ShortArray, IntArray, LongArray, FloatArray, DoubleArray, CharArray, BooleanArray

연산자

연산자와 연산자 오버로딩

- 내부적으로 연산자 오버로딩(overloading)한 함수를 사용해서 연산을 처리

표기	변환 코드	의미
a + b	a.plus(b)	a와 b의 값을 더한다
a - b	a.minus(b)	a의 값에서 b의 값을 뺀다
a * b	a.times(b)	a의 값과 b의 값을 곱한다
a / b	a.div(b)	a의 값을 b의 값으로 나눈다.
a % b	a.rem(b), a.mod(b)	a의 값을 b의 값으로 나눈 후 나머지를 구한다

Byte 클래스의 plus 연산자

```
/** Adds the other value to this value. */
public operator fun plus(other: Byte): Int
/** Adds the other value to this value. */
public operator fun plus(other: Short): Int
/** Adds the other value to this value. */
public operator fun plus(other: Int): Int
/** Adds the other value to this value. */
public operator fun plus(other: Long): Long
/** Adds the other value to this value. */
public operator fun plus(other: Float): Float
/** Adds the other value to this value. */
public operator fun plus(other: Double): Double
```

연산자와 연산자 오버로딩

- 단항 연산자

표기	변환 코드	의미
<code>+a</code>	<code>a.unaryPlus()</code>	a의 값을 양수로 변환
<code>-a</code>	<code>a.unaryMinus()</code>	a의 값을 음수로 변환
<code>!a</code>	<code>a.not()</code>	Boolean 타입의 부정(true는 false로, false는 true로)
<code>++a, a++</code>	<code>inc</code>	a의 값에 1을 더함
<code>--a, a--</code>	<code>dec</code>	a의 값에서 1을 뺌

- 복합 대입 연산자

표기	변환 코드	의미
<code>a += b</code>	<code>a.plusAssign(b)</code>	a의 값에 b의 값을 더한 후 a에 넣음
<code>a -= b</code>	<code>a.minusAssign(b)</code>	a의 값에서 b의 값을 뺀 후 a에 넣음
<code>a *= b</code>	<code>a.timesAssign(b)</code>	a의 값에 b의 값을 곱한 후 a에 넣음
<code>a /= b</code>	<code>a.divAssign(b)</code>	a의 값을 b의 값으로 나눈 후 a에 넣음
<code>a %= b</code>	<code>a.modAssign(b)</code>	a의 값을 b의 값으로 나눈 후 나머지를 a에 넣음

연산자와 연산자 오버로딩

- 비트 연산자 – 오버로딩하고 있지 않음

함수명	의미
shl	부호 비트는 그대로 두고 왼쪽으로 비트 이동(Signed shift left)
shr	부호 비트는 그대로 두고 오른쪽으로 비트 이동(Signed shift right)
ushr	부호 비트를 포함해서 오른쪽으로 비트 이동(Unsigned shift right)
and	대응되는 각 비트에 대해 논리 and 연산 수행(Bitwise and)
or	대응되는 각 비트에 대해 논리 or 연산 수행(Bitwise or)
xor	대응되는 각 비트에 대해 논리 xor(exclusive or) 연산 수행(Bitwise xor)
inv	0또는 1의 비트 값을 반대로 바꿈(Bitwise inversion)

```
/** Shifts this value left by the [bitCount] number of bits.  
public infix fun shl(bitCount: Int): Int
```

```
println(8 shr 2)  
println(8 shl 4)  
println(0xC0 and 0x0C)  
println(0xC0 or 0x0C)  
println(0xC0 xor 0x0C)
```

연산자와 연산자 오버로딩

- 논리 연산자

함수명	의미	
and	좌우의 피 연산자에 대해 논리 and 연산 수행	&&
or	좌우의 피 연산자에 대해 논리 or 연산 수행	
not	좌우의 피 연산자에 대해 논리 not 연산 수행	!

```
val b1:Boolean = true
```

```
val b2:Boolean = false
```

```
println(b2 and b1)
```

```
println(b1 or b2)
```

```
println(b1 and b2.not())
```

단락평가(x)

```
val b1:Boolean = true
```

```
val b2:Boolean = false
```

```
println(b2 && b1)
```

```
println(b1 || b2)
```

```
println(b1 && !b2)
```

단락평가(O)

연산자와 연산자 오버로딩

- 동등 비교 연산자

표기	변환 코드
<code>a == b</code>	<code>a?.equals(b) ?: (b == null)</code>
<code>a === b</code>	오버로딩 안 됨 변수의 참조값 비교 (똑같은 객체를 참조하는지)
<code>a != b</code>	<code>!(a?.equals(b) ?: (b == null))</code>
<code>a !== b</code>	오버로딩 안 됨
<code>not</code>	좌우의 피 연산자에 대해 논리 not 연산 수행

- 기타 연산자

표기	변환 코드
<code>a > b</code>	<code>a.compareTo(b) > 0</code>
<code>a < b</code>	<code>a.compareTo(b) < 0</code>
<code>a >= b</code>	<code>a.compareTo(b) >= 0</code>
<code>a <= b</code>	<code>a.compareTo(b) <= 0</code>

연산자와 연산자 오버로딩

- in 연산자
 - 특정 객체가 컬렉션 (List 나 Map)에 저장되어 있는지 확인

표기	변환 코드
<code>a in b</code>	<code>b.contains(a)</code>
<code>a !in b</code>	<code>!b.contains(a)</code>

- 범위(..) 연산자
 - `a..b` : a에서 b까지의 범위 값이 생성 ➔ `a.rangeTo(b)` 코드로 변환

```
val start = LocalDate.now()
```

```
val end = start..start.plusDays(15)    *Closed range
```

```
println(start.plusWeeks(1) in end)
```

```
println(end)
```

```
2024-03-02 .. 2024-03-17
```

연산자와 연산자 오버로딩

- 인덱스 연산자

표기	변환 코드
a[i]	a.get(i)
a[i, j]	a.get(i, j)
a[i_1, ..., i_n]	a.get(i_1, ..., i_n)
a[i] = b	a.set(i, b)
a[i, j] = b	a.set(i, j, b)
a[i_1, ..., i_n] = b	a.set(i_1, ..., i_n, b)

- invoke 연산자

- 객체를 통한 함수 호출 (인스턴스를 함수 처럼 사용)

```
class InvokeOperator(val makeMessage1: String) {  
    operator fun invoke(makeMessage2: String) {  
        println("$makeMessage1 $makeMessage2!")  
    }  
}
```

```
val instance1 = InvokeOperator("코틀린을")  
instance1("배우자")
```

연산자와 연산자 오버로딩

- 타입 확인 연산자 : is, !is

```
val b: String = "코틀린을 배우자"
if (b is String) {
    println("String 타입임")
} else {
    println("String 타입이 아님")
}
```

난수 발생

- Java의 Random 클래스 활용 가능
 - Java.util.random 클래스 import 해야 함

```
val num = Random().nextInt(10)
```

- Kotlin

```
val num = (0..10).random()
```

입출력

출력

- 출력

- println, print 함수 사용 → 자바와 동일
- 출력 문자열 내에 \$는 변수 값 출력

```
val score = 12.3 println("score")
```

```
println("$score") println("score = $score")
```

```
println("${score + score}") println(12.3)
```

score

12.3

score = 12.3

24.6

12.3

입력

- String 입력 : readln 함수 ➔ 다른 타입은 형변환

```
print("Enter text: ")  
val stringInput = readln()  
println("You entered: $stringInput")
```

- 다른 타입의 데이터 입력
 - 자바의 Scanner 객체 사용 (import java.util.Scanner)

```
val reader = Scanner(System.`in`)  
print("Enter a number: ")  
var integer:Int = reader.nextInt()  
println("You entered: $integer")
```


제어문

제어문 - if문

- If 문

- 기존 java와 같은 형태로 사용 가능함

```
if(a > b){  
    result = a  
}else{  
    result = b  
}
```

- 코틀린에서는 if문을 하나의 표현식으로 간주하며, 값을 반환함.

- 따라서, 변수의 대입문에 식처럼 사용 가능

```
val result = if(a>b) a else b
```

```
val number = -10  
val result = if (number > 0)  
    "Positive number"  
else  
    "Negative number"
```

제어문 - if문

- If문

- 조건에 따라 여러 표현식을 실행해야 하는 경우, **블록의 마지막 값이 반환됨**

```
val a = -9
val b = -11
val max = if (a > b) {
    println("$a is larger than $b.")
    println("max variable holds value of a.")
    a
} else {
    println("$b is larger than $a.")
    println("max variable holds value of b.")
    b
}
```

제어문 - when문

- when 문
 - Switch 구문에 대응하는 구문으로 break 구문 필요 없음

```
val a = 12
```

```
val b = 5
```

```
println("Enter operator either +, -, * or /")
```

```
val op = readln()
```

```
val result = when (op) {
```

```
    "+" -> a + b
```

```
    "-" -> a - b
```

```
    "*" -> a * b
```

```
    "/" -> a / b
```

```
    else -> "$op operator is invalid operator."
```

```
}
```

제어문 - when문

- when 문
 - Switch 구문에 대응하는 구문

```
val x = 9
when(x){
    1 -> println("x==1")
    2,3 -> println("x==2 or x==3")
    in 4..7 -> println("4부터 7사이")
    !in 8..10 -> println("8부터 10사이가 아님")
    else ->{
        print("x는 8에서 10사이의 숫자임")
    }
}
```

제어문 - when문

- when 문
 - 변수에 대입 가능

```
val number = 1
val numStr = when(number % 2){
    0->"짝"
    else->"홀"
}
println(numStr)
```

- 함수의 반환 값으로 사용 가능

```
fun isEven(num:Int) : String = when(num%2){
    0->"짝"
    else->"홀"
}
println( isEven(100) )
```

```
fun isEven(num:Int) : String{
    return when(num%2){
        0->"짝"
        else->"홀"
    }
}
```

반환 타입 생략 가능

```
fun isEven(num:Int) = when(num%2){...}
```

제어문 - for문

- For 문

```
val numbers = arrayOf(1,2,3,4,5)
for(num in numbers){
    println(num)
}
```

```
val numbers = arrayOf(1,2,3,4,5)
for(index in numbers.indices){
    println("number at $index is ${numbers[index]}")
}
```

```
for(i in 1..3){}
```

```
for(i in 0..10 step 2){}
```

```
for(i in 10 downTo 0 step 2){}
```

```
for(i in 1 until 10){} // 1씩 증가
```

```
repeat(10){}
```

제어문 - foreach

- 각 원소에 대해 반복 처리시 유용

```
val names = arrayOf("greenjoa1", "greenjoa2", "greenjoa3", "greenjoa4")  
names.forEach {name->  
    println(name)  
}
```


제어문 - while문

- While 과 do-while 문

```
var x=10
while(x>0){
    println(x)
    x--
}
```


```
val items = listOf("사과", "바나나", "키위")
var index = 0
while(index < items.size){
    println("item at $index is ${items[index]}")
    index++
}
```

```
var x=10
do{
    println(x)
    x--
}while(x>0)
```


```
val items = listOf("사과", "바나나", "키위")
var index = 0
do{
    println("item at $index is ${items[index]}")
    index++
}while(index < items.size)
```

제어문 - Break


```
while (testExpression) {  
    // codes  
    if (condition to break) {  
        break  
    }  
    // codes  
}
```



```
do {  
    // codes  
    if (condition to break) {  
        break  
    }  
    // codes  
} while (testExpression)
```




```
for (iteration through iterator) {  
    // codes  
    if (condition to break) {  
        break  
    }  
    // codes  
}
```



제어문 - Break

- Labeled break

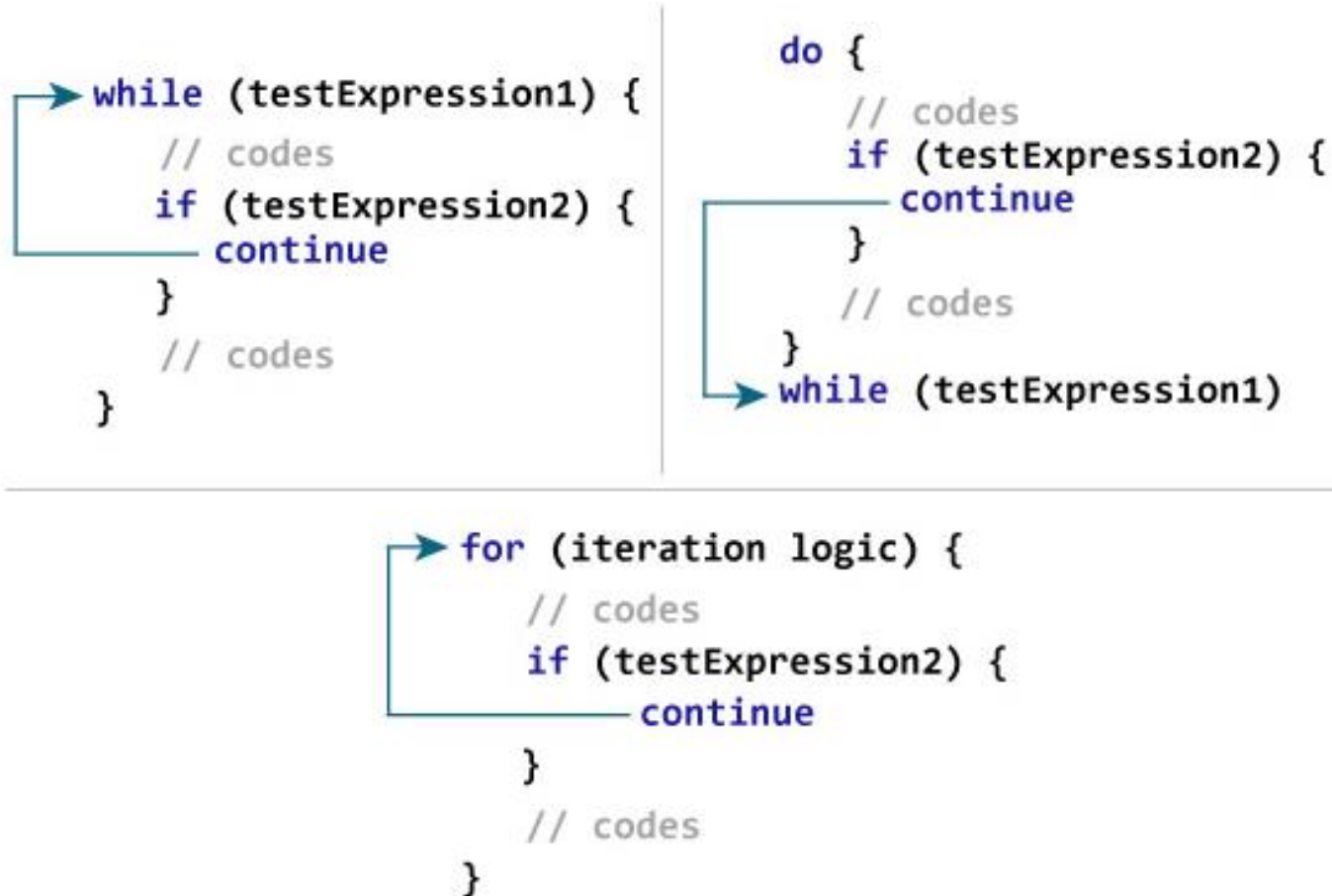
```
test@ while (testExpression) {  
    // codes  
    while (testExpression) {  
        // codes  
        if (condition to break) {  
            break@test  
        }  
        // codes  
    }  
    // codes  
}
```



```
first@ for (i in 1..4) {  
    second@ for (j in 1..2) {  
        println("i = $i; j = $j")  
        if (i == 2)  
            break@first  
    }  
}
```

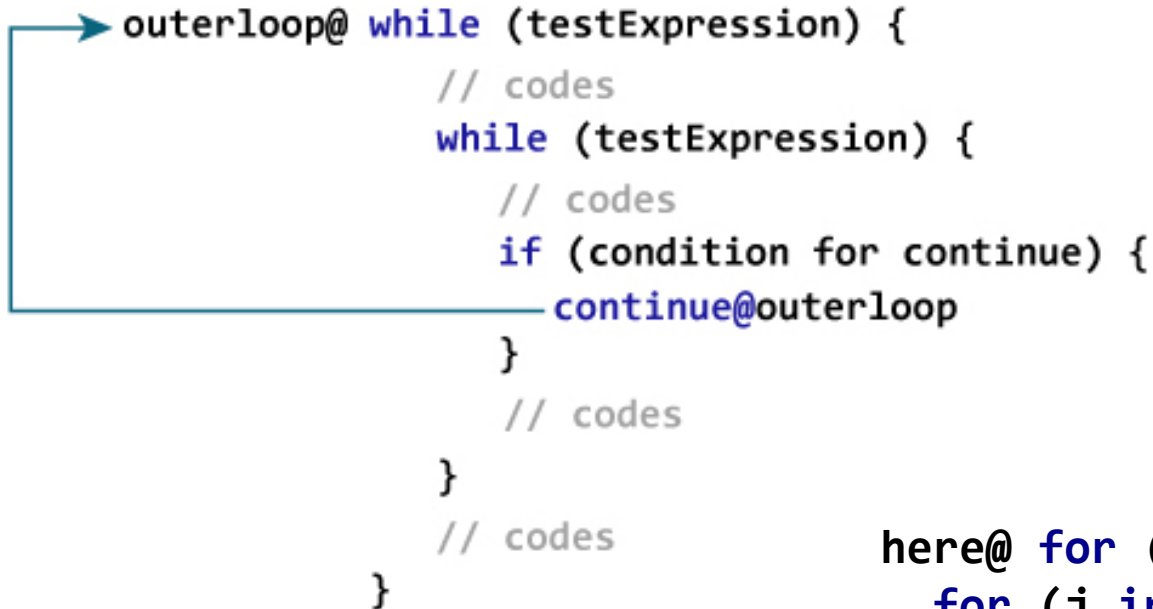
*공백있으면 안됨

제어문 - Continue



제어문 - Continue

- Labeled Continue



```
outerloop@ while (testExpression) {  
    // codes  
    while (testExpression) {  
        // codes  
        if (condition for continue) {  
            continue@outerloop  
        }  
        // codes  
    }  
    // codes  
}
```

```
here@ for (i in 1..5) {  
    for (j in 1..4) {  
        if (i == 3 || j == 2)  
            continue@here  
        println("i = $i; j = $j")  
    }  
}
```

수고하셨습니다.