방구석 개발자 2탄

알기 쉬운 **안드로이드 앱 기초**

처음부터 배우는
안드로이드 앱 개발 기초부터 실전까지

본 교재는 2025년도 과학기술정보통신부 및 정보통신기획평가원의 'SW중심대학사업' 지원을 받아 제작되었습니다. (2023-0-00076) 본 결과물의 내용을 전재할 수 없으며, 인용(재사용)할 때에는 반드시 과학기술정보통신부와 정보통신기획평가원이 지원한 'SW중심대학'의 결과물이라는 출처를 밝혀야 합니다.

알기 쉬운 안드로이드 앱 기초

처음부터 배우는
안드로이드 앱 개발 기초부터 실전까지

발행일 : 2025년 12월 15일
지은이 : 박경훈, 주동현, 김영준
발행인 : 박지연
발행처 : 라임하우스
주 소 : 서울시 금천구 범안로 1142, 더스카이밸리 2차 402호
서울시 영등포구 선유로 70, 409호
전 화 : 02-857-9984
팩 스 : 0504-202-0920
E-mail : jwj3207@gmail.com

Copyright©라임하우스, 2025, Printed in Korea
ISBN : 979-11-
정 가 : 18,000

※ 잘못된 책은 교환하여 드립니다.
※ 이 책은 저작권법에 따라 보호 받는 저작물이므로 무단전제와 무단복제를 금지하며, 책의 내용, 이미지, 편집 구성 등을 이용하려면 반드시 저작권자와 라임하우스의 서면동의를 받아야합니다.

방구석 개발자 2탄

알기 쉬운
안드로이드 앱 기초

처음부터 배우는
안드로이드 앱 개발 기초부터 실전까지

박경훈, 주동현, 김영준 지음



라임하우스

저자소개

박경훈

- 신호처리(음성), 데이터베이스, 임베디드 시스템 전문가
- 동아대학교 전자공학과 박사수료
- 현) 동아대학교 소프트웨어혁신센터 교수

주동현

- AI 기반 신호처리 / 데이터베이스 / 임베디드 시스템 전문가
- 동아대학과 전자공학과 박사
- 현) 한국폴리텍대학 부산캠퍼스 소프트웨어융합과 교수

김영준

- 모바일 네트워크, 엣지컴퓨팅, 강화학습 등 연구자
- 고려대학교 전기전자전파공학과 박사
- 현) 경남대학교 컴퓨터공학부 조교수

저자 서문

모바일 산업의 급속한 발전과 함께 스마트 기기의 활용이 일상화되면서, 모바일 애플리케이션 개발에 대한 수요는 지속적으로 증가하고 있습니다. 그 중에서도 안드로이드 운영체제는 전 세계적으로 널리 보급된 플랫폼으로, 이에 대한 이해와 개발 역량은 소프트웨어 기초 교육에 있어 필수적인 요소로 자리 잡고 있습니다.

본 교재『안드로이드 앱 기초』는 안드로이드 앱 개발에 처음 입문하는 학습자들을 대상으로 구성되었으며, 기본 개념부터 실습 중심의 학습까지 단계적으로 학습할 수 있도록 내용을 체계화하였습니다. 개발 환경 설정, 앱의 기본 구조, 이벤트 처리, 메뉴와 대화상자 구성, 애니메이션 구현, 액티비티와 인텐트의 활용, 리스트뷰와 어댑터뷰 등 안드로이드 개발의 핵심 주제를 포함하고 있으며, 마지막 장에서는 공공데이터를 활용한 간단한 앱 개발 실습을 통해 학습한 내용을 종합적으로 적용해볼 수 있도록 하였습니다.

본 교재는 개념 설명과 실습 예제를 병행하여 학습 효과를 극대화하는 데 중점을 두었으며, 초급 수준의 실습 경험을 통해 학습자가 실무 개발에 대한 기초 역량을 갖출 수 있도록 구성하였습니다. 향후 보다 고급 수준의 개발 학습으로 나아가기 위한 기초 다지기의 기회가 되기를 기대합니다.

학습자 여러분이 본 교재를 통해 안드로이드 앱 개발에 대한 흥미를 갖고, 정보기술 시대의 핵심 인재로 성장하는 데 밑거름이 되기를 바랍니다.

2025년 11월
저자 일동

목차

저자 서문 5

1장 안드로이드 앱의 이해와 개발 환경 설정 9

1.1 안드로이드 앱이란? 10

1.1.1 안드로이드 앱의 특징 10

1.1.2 안드로이드 앱의 구성 요소 10

1.2 안드로이드의 역사 11

1.3 안드로이드 개발 환경 설정 12

1.3.1 안드로이드 스튜디오(Android Studio) 설치 12

1.3.2 SDK(Software Development Kit) 설정 18

1.3.3 AVD(Android Virtual Device) 생성 21

2장 안드로이드 앱의 기본 구조와 인터페이스 27

2.1 안드로이드 앱 기본 구조 28

2.1.1 프로젝트 뷰(Project View) 살펴보기 29

2.1.2 manifest -> AndroidManifest.xml 30

2.1.3 Gradle Scripts 32

2.2 인터페이스 기초 34

2.2.1 액티비티(Activity)와 레이아웃(Layout) 35

2.2.2 뷰(View)와 위젯(Widget) 38

2.2.3 XML 레이아웃과 코드 연결하기 40

2.3 Layout 실습 42

2.3.1 LinearLayout 실습 42

2.3.2 RelativeLayout 실습 46

3장 이벤트(Event) 53

3.1 익명 클래스(Anonymous Class) 54

3.1.1 익명 클래스란? 55

3.1.2 View.OnClickListener 인터페이스와 익명 클래스 55

3.1.3 익명 클래스 사용의 장단점 59
3.2 익명 클래스 참조 객체 생성 61
3.2.1 참조 객체 생성 방법 61
3.2.2 익명클래스 참조 객체 장점 및 활용 63
3.3 인터페이스 상속 (Activity에 구현) 64
3.3.1 Activity에 View.OnClickListener 구현하기 64
3.3.2 장단점 및 활용 67
3.4 위젯 onClick 속성 활용 67
3.4.1 android:onClick 속성 사용 방법 68
3.4.2 위젯 onClick 장점 및 단점 71
3.4.3 언제 android:onClick을 사용할까? 71
3.5 이벤트 처리 방법 선택 가이드 72

4장 메뉴와 대화상자 75
4.1 메뉴 76
4.1.1 옵션 메뉴(Option Menu) 77
4.1.2 메뉴(Menu) 활성화 84
4.1.3 컨텍스트 메뉴(Context Menu) 94
4.1.4 팝업 메뉴(Popup Menu) 98
4.2 대화상자(Dialog) 106
4.2.1 AlertDialog (경고 대화상자) 107
4.2.3 날짜와 시간 Dialog (DatePickerDialog, TimePickerDialog) 120

5장 애니메이션 125
5.1 XML 활용 애니메이션 126
5.1.1 애니메이션 개요 126
5.1.2 View Animation (트윈 애니메이션) 127
5.1.3 Drawable Animation (프레임 애니메이션) 133
5.1.4 Interpolator 138
5.2 코딩 활용 애니메이션 145
5.2.1 Property Animation 146
5.2.2 ObjectAnimator Animation 150

6장 액티비티와 인텐트 155
6.1 액티비티와 인텐트 기본 156
6.1.1 액티비티(Activity) 156
6.1.2 인텐트 (Intent) 161
6.2 액티비티와 인텐트 실습 예제 167
6.2.1 명시적 인텐트: 화면 전환 + parcelable 전달 167
6.2.2 암시적 인텐트 (웹 / 통화 / 지도 / 공유) 173
6.2.3 카메라 177

7장 어댑터 뷰 181
7.1 Adapter 182
7.1.1 Adapter 역할 182
7.1.2 Adapter 동작 원리 183
7.1.3 Adapter 종류 184
7.2 Adapter 실습 192
7.2.1 리스트(List) 뷰 192
7.2.2 리스트(List) 뷰 응용 196

8장 앱 개발 217
8.1 공공 데이터 활용방법 218
8.1.1 공공데이터 사전 준비 218
8.2 공공데이터 활용 앱 개발 219
8.2.1 API 검색 및 활용 방법 220
8.2.2 공공데이터 활용한 앱 개발 230

맺음말 253



제 1 장

안드로이드 앱의 이해와 개발 환경 설정



학습목표

- Android Studio, Android SDK를 설치 및 구성하고, AVD(에뮬레이터) 또는 실기기를 연결한다.
- 안드로이드 플랫폼을 이해하고, 주요 구성요소를 이해한다.

이 챕터에서는 안드로이드 앱 개발을 시작하기 위한 기본적인 개념과 필요한 개발 환경 설정 방법을 다룹니다.

1.1 안드로이드 앱이란?

안드로이드 앱은 구글에서 개발한 안드로이드 운영체제 위에서 동작하는 소프트웨어 애플리케이션을 의미합니다. 전 세계 스마트폰의 대부분을 차지하는 안드로이드는 사용자들에게 다양한 기능을 제공하며, 안드로이드 앱은 이러한 기능들을 구현하는 핵심 요소입니다.

1.1.1 안드로이드 앱의 특징

- 오픈 소스 플랫폼: 안드로이드 운영체제는 오픈 소스로 공개되어 있어 누구나 자유롭게 개발하고 배포할 수 있습니다. 이는 개발자들에게 높은 자유도를 제공하며, 다양한 앱 생태계를 형성하는 데 기여
- 다양한 기기 지원: 스마트폰뿐만 아니라 태블릿, 스마트 TV, 웨어러블 기기, 자동차 등 다양한 안드로이드 기반 기기에서 앱 실행 가능
- 자바(Java) 및 코틀린(Kotlin) 기반: 주로 자바와 코틀린 프로그래밍 언어를 사용하여 개발됩니다. 최근에는 구글에서 공식적으로 코틀린을 안드로이드 앱 개발의 주요 언어로 권장

  단, 본 교재에서는 Java 기반 안드로이드 앱 개발로 진행
- 구글 플레이 스토어: 개발된 앱은 구글 플레이 스토어를 통해 전 세계 사용자들에게 쉽게 배포하고 판매 가능

1.1.2 안드로이드 앱의 구성 요소

안드로이드 앱은 여러 가지 구성 요소가 유기적으로 연결되어 동작합니다. 주요 구성 요소는 다음과 같습니다.

- 액티비티(Activity): 사용자 인터페이스(UI)를 담당하며, 사용자와 상호작용하는 화면 하나하나를 나타냅니다. 예를 들어, 앱을 실행했을 때 보이는 첫 화면이나 특정 버튼을 눌렀을 때 나타나는 화면 등이 액티비티에 해당합니다.
- 서비스(Service): 백그라운드에서 실행되는 작업을 처리할 때 사용됩니다. 사용자 인터페이

스 없이 음악 재생, 네트워크 데이터 전송 등 오랜 시간 동안 실행되어야 하는 작업을 수행합니다.

- 콘텐츠 프로바이더(Content Provider): 앱 간에 데이터를 공유할 수 있도록 해주는 구성 요소입니다. 예를 들어, 연락처 앱의 데이터를 다른 앱에서 접근하여 사용할 수 있도록 합니다.
- 브로드캐스트 리시버(Broadcast Receiver): 시스템에서 발생하는 다양한 이벤트(배터리 부족, 전화 수신, 부팅 완료 등)를 수신하고 반응하는 역할을 합니다.

1.2 안드로이드의 역사

안드로이드는 2003년 앤디 루빈, 리치 마이너, 닉 시어스, 크리스 화이트가 설립한 안드로이드 주식회사(Android Inc.)에서 시작되었습니다. 초기에는 디지털 카메라용 운영체제를 목표로 했으나, 시장성 문제로 인해 스마트폰 운영체제로 방향을 전환했습니다.

표 1-1 안드로이드 연혁

연도	주요 내용
2003	안드로이드 주식회사 설립
2005	구글, 안드로이드 주식회사 인수
2007	OHA(Open Handset Alliance) 결성, 안드로이드 오픈 소스 프로젝트(AOSP) 발표
2008	최초의 안드로이드 스마트폰 G1(HTC Dream) 출시, 안드로이드 1.0 정식 발표
2009	안드로이드 1.5 컵케이크(Cupcake) 출시 (버전별 디저트 이름 시작)
2010	안드로이드 2.2 프로요(Froyo) 출시, 삼성 갤럭시 S 출시
2011	안드로이드 3.0 허니콤(Honeycomb) 출시 (태블릿 전용)
2012	안드로이드 4.1 젤리빈(Jelly Bean) 출시
2013	안드로이드 4.4 킷캣(KitKat) 출시
2014	안드로이드 5.0 롤리팝(Lollipop) 출시
2015	안드로이드 6.0 마시멜로(Marshmallow) 출시
2016	안드로이드 7.0 누가(Nougat) 출시

2017	안드로이드 8.0 오레오(Oreo) 출시
2018	안드로이드 9 파이(Pie) 출시
2019	안드로이드 10 출시 (디저트 이름 폐지)
2020	안드로이드 11 출시
2021	안드로이드 12 출시
2022	안드로이드 13 출시
2023	안드로이드 14 출시
2024	안드로이드 15 출시 (예정)

안드로이드는 꾸준한 업데이트를 통해 기능과 성능을 개선하고 있으며, 개발자들에게 더욱 강력한 도구와 기능을 제공하고 있습니다.

1.3 안드로이드 개발 환경 설정

안드로이드 앱을 개발하기 위해서는 특정 소프트웨어와 개발 도구가 필요합니다. 이 섹션에서는 안드로이드 스튜디오를 설치하고 개발 환경을 설정하는 방법을 단계별로 안내합니다.

1.3.1 안드로이드 스튜디오(Android Studio) 설치

안드로이드 스튜디오는 구글에서 공식적으로 제공하는 안드로이드 앱 개발을 위한 통합 개발 환경(IDE)입니다. 자바와 코틀린 코드를 작성하고, UI를 디자인하며, 앱을 빌드하고 디버깅하는 데 필요한 모든 기능을 제공합니다.

- 시스템 요구 사항 확인:
  안드로이드 스튜디오를 설치하기 전에 컴퓨터가 최소 시스템 요구 사항을 충족하는지 확인하세요. 일반적으로 8GB 이상의 RAM과 8GB 이상의 하드 디스크 여유 공간이 권장됩니다.
- 안드로이드 스튜디오 다운로드:
  ① 웹 브라우저를 열고 https://developer.android.com/studio 에 접속합니다.

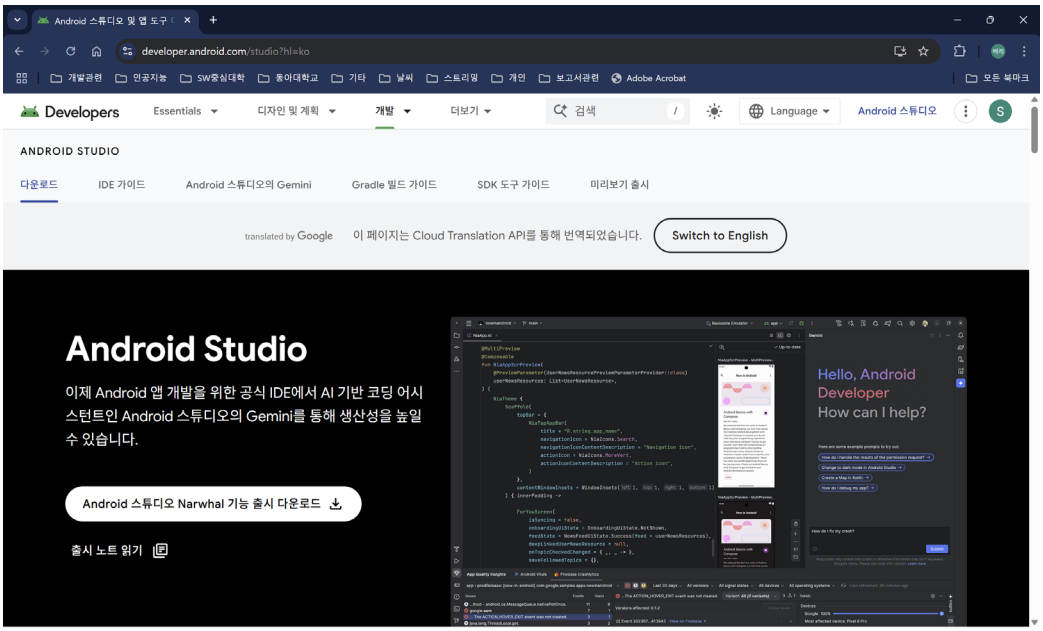


그림 1-1 ANDROID STUDIO 개발자 웹사이트 접속화면

② "Android 스튜디오 Narwhal 기능 출시 다운로드" 버튼을 클릭하여 최신 버전의 안드로이드 스튜디오 설치 파일을 다운로드합니다.

다운로드 약관에 동의하고 설치를 진행합니다. 운영체제(Windows, macOS, Linux)에 맞는 버전을 선택해야 합니다.

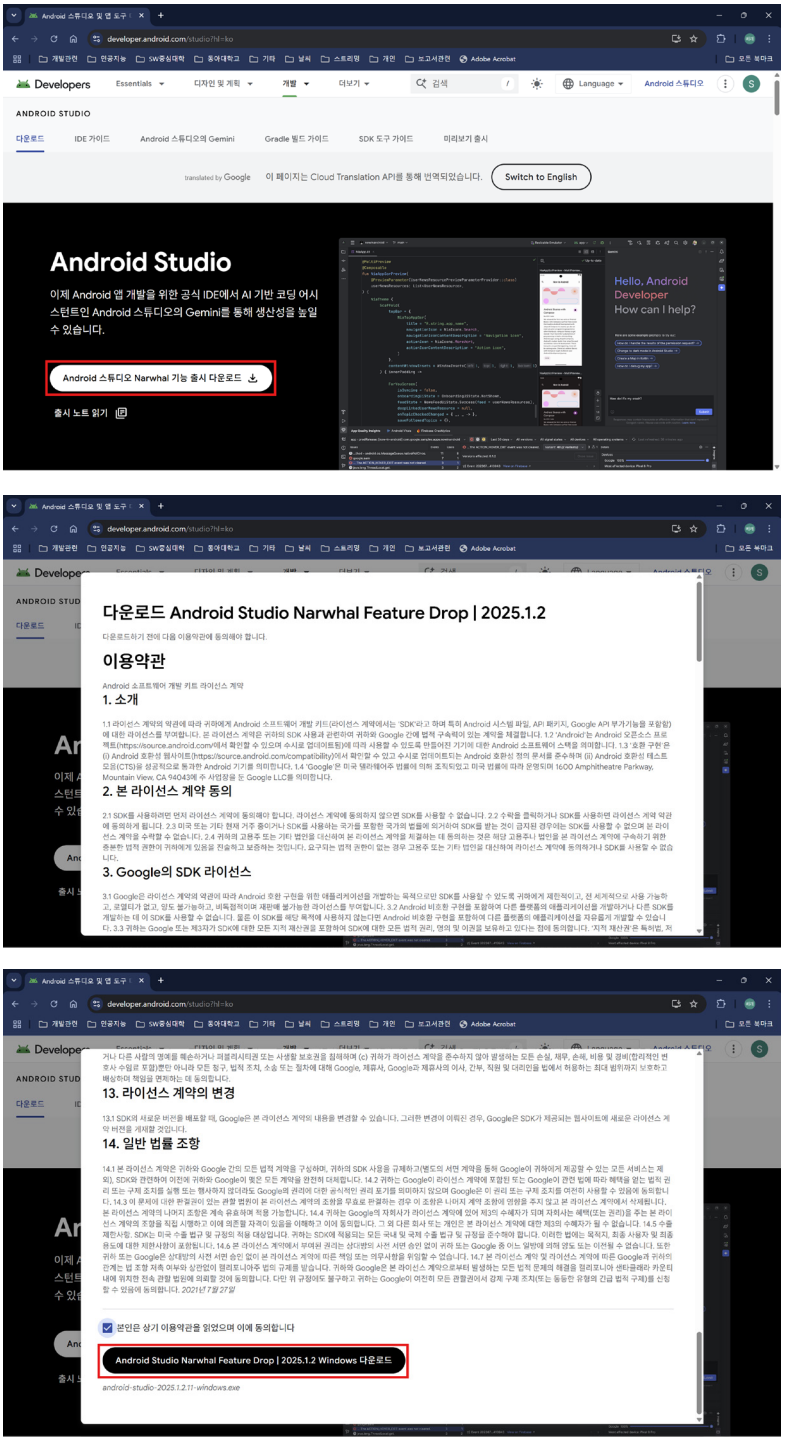


그림 1-2 ANDROID STUDIO 이용약관 동의체크 및 다운로드 클릭

- 설치 과정:

  다운로드한 설치 파일을 실행한 후 설치 마법사의 지시에 따라 설치를 진행합니다. 기본 설정으로 설치하는 것을 권장합니다.

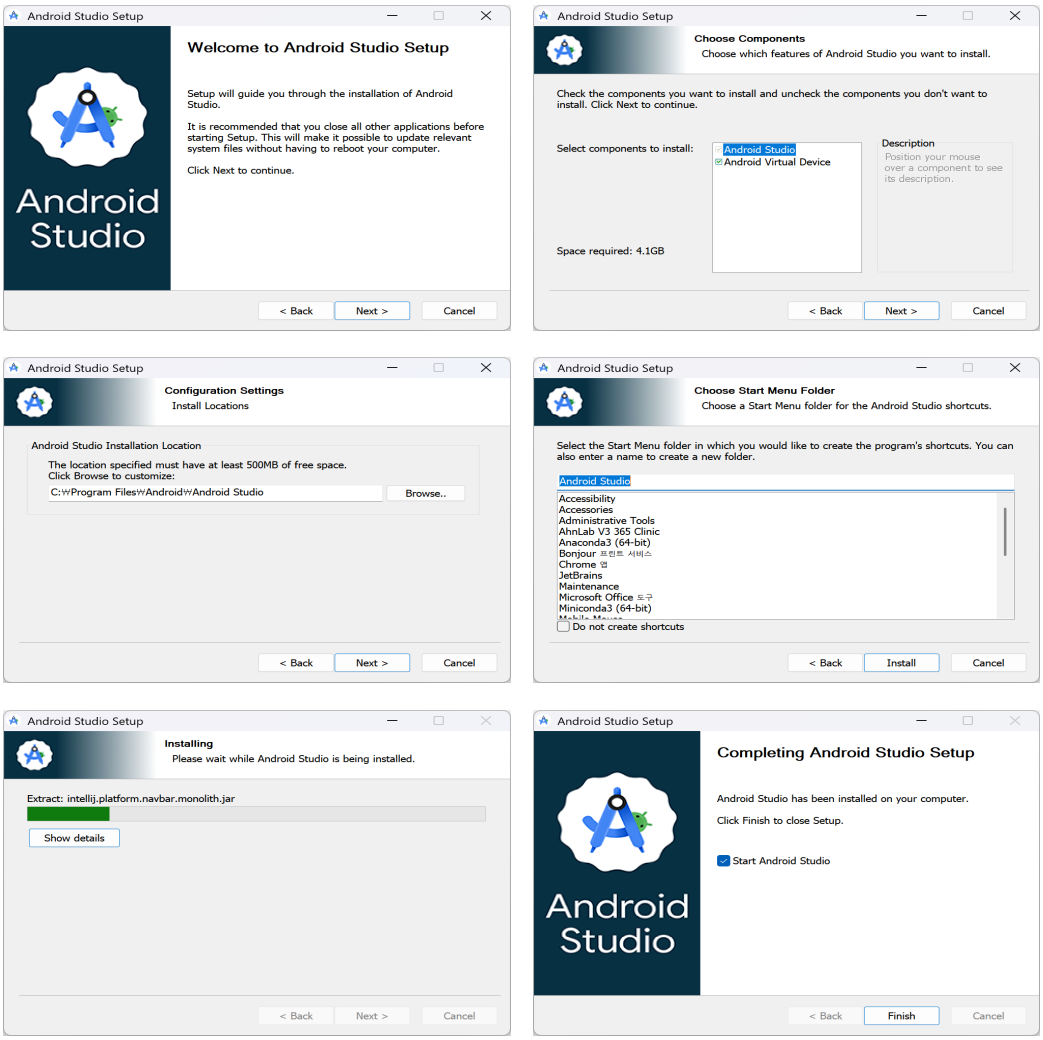


그림 1-3 Android Studio 단계별 설치화면

Android Studio 설치 완료 후 Android Studio 성능개선을 위한 정보제공 요청이 나오는데 Android Stduio 설치와 실행과 무관하므로 [Don`t send] 를 선택하면 된다.

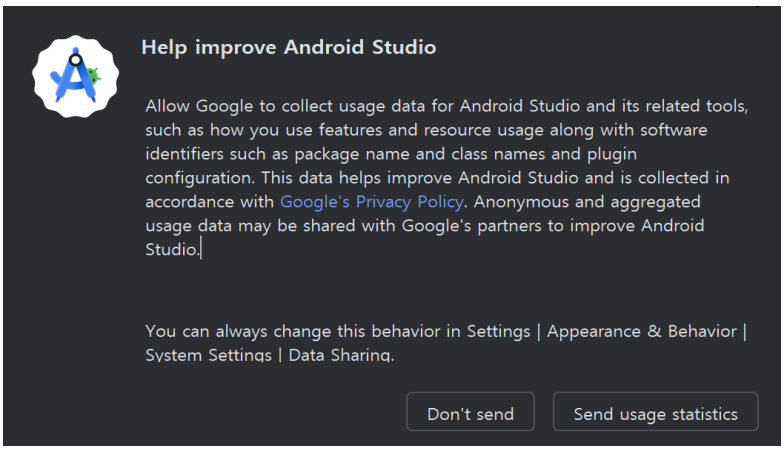
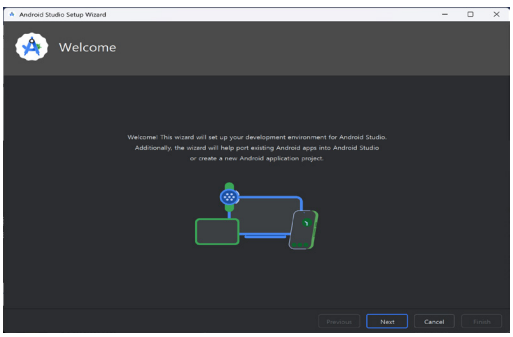
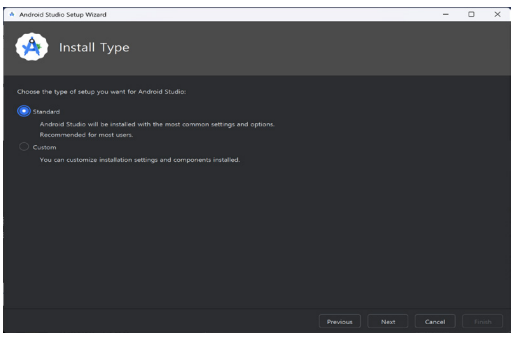


그림 1-4 Android Studio 성능개선 정보제공 요청화면

아래는 Android Studio 셋업 환경을 설정하는 단계이다.





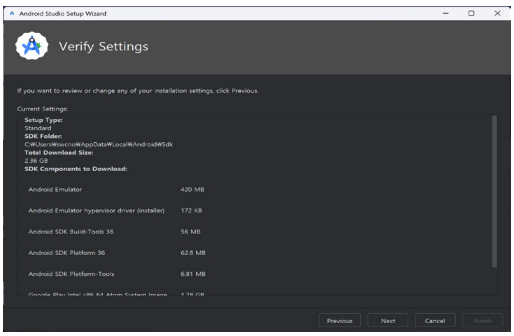


그림 1-5 Android Studio 셋업 환경설정 화면

다음으로 Android SDK(Software Development Kit), Android Virtual Device(AVD) 등의 구성 요소를 다운로드 한다.

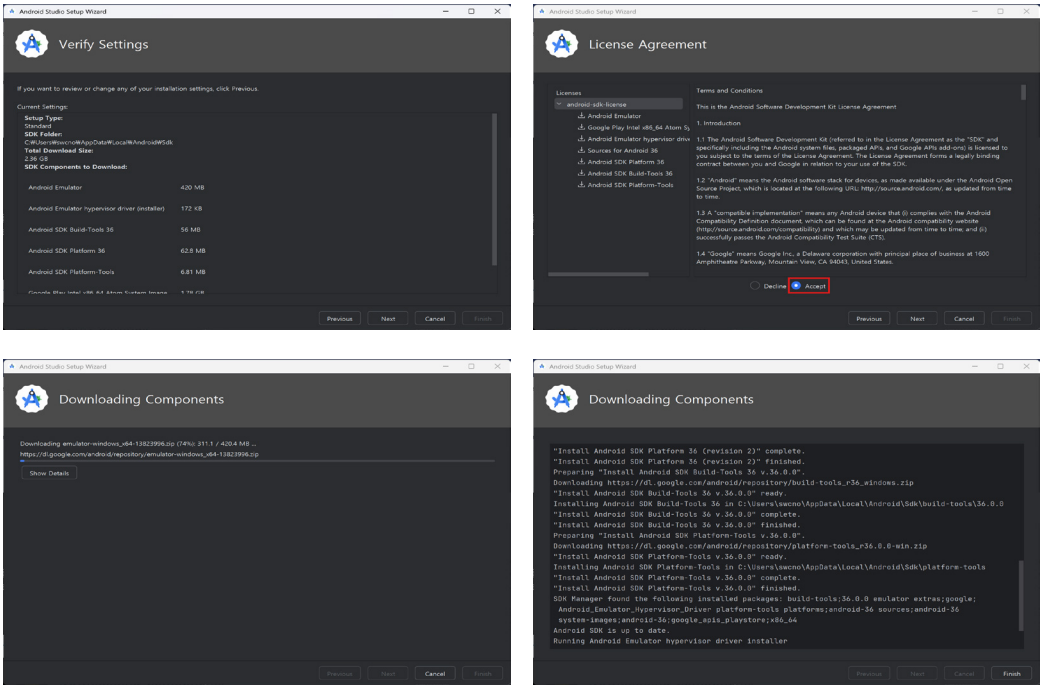


그림 1-6 Android SDK, AVD 등 다운로드 및 설치 화면

[Finish] 버튼을 클릭하여 완료한다.

1.3.2 SDK(Software Development Kit) 설정

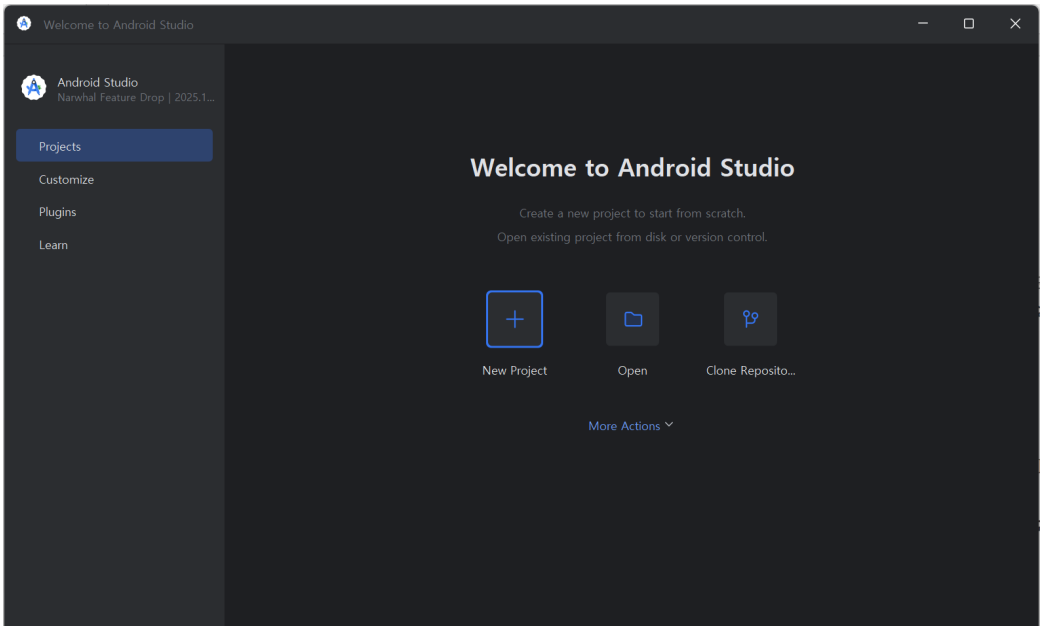


그림 1-7 Android Studio Welcome 화면

안드로이드 스튜디오를 설치 모든 과정을 완료한 후 "안드로이드 스튜디오 시작화면(Welcome Screen)"이 나타나게 된다. 앞으로 자주 보게 될 화면이며, 새로운 프로젝트, 기존 작업 프로젝트 관리를 할 수 있는 화면이다. 본 화면에서 수행할 수 있는 주요 작업은 아래와 같습니다.

- New Project : 새로운 안드로이드 앱 프로젝트 생성
- Open : 기존에 작업하던 프로젝트 폴더를 열어 작업
- Clone Repository : Git과 같은 버전 관리 시스템의 저장소에서 프로젝트 로드 후 작업

위에서 언급한 작업 이외에도 Android 앱 개발환경 설정을 위한 [SDK Manager]를 통해 SDK 구성 요소 관리, 버전별 SDK 플랫폼 관리, 개발도구 관리 및 최신화(업데이트) 작업을 수행할 수 있고, 설정할 수 있습니다.

- SDK Manager : [More Actions] 〉 SDK Manager
- SDK Platforms 탭: 안드로이드 버전별 SDK 플랫폼을 관리

최신 안정화된 버전의 안드로이드 SDK 플랫폼(예: Android 34, Android 33)을 선택하고"Apply" 버튼을 클릭하여 다운로드 및 설치합니다. 다양한 버전의 SDK를 설치해두면, 여러 안드로이드 버전에서 앱을 테스트할 수 있어 편리하게 활용할 수 있습니다.

- SDK Tools 탭: SDK 개발 도구 관리

Android SDK Build-Tools, Android SDK Platform-Tools, Android Emulator 등 중요한 도구들이 최신 버전으로 설치되어 있는지 확인합니다. 업데이트가 필요한 경우 체크박스를 선택하고 "Apply" 버튼을 클릭합니다.



주요 개념

'안드로이드 플랫폼'은 안드로이드 앱 개발의 기반이 되는 소프트웨어 전체를 일컫는 말이며, 단순히 '안드로이드 운영체제'만을 의미하는 것이 아니라, 앱이 원활하게 작동할 수 있도록 지원하는 모든 구성 요소를 포괄하는 개념입니다.
안드로이드 플랫폼은 여러 계층으로 이루어진 소프트웨어 스택(Software Stack) 구조를 가지고 있습니다. 이 스택은 크게 다음과 같은 주요 구성 요소로 나눌 수 있습니다.

① 리눅스 커널 (Linux Kernel):

- 안드로이드 플랫폼의 가장 밑바닥에 위치하는 핵심 계층입니다.
- 메모리 관리, 프로세스 관리, 하드웨어 드라이버와 같은 기본적인 시스템 기능들을 제공합니다.
- 안드로이드는 리눅스 커널을 기반으로 하기 때문에, 안정성과 보안, 멀티태스킹 등의 강력한 기능을 활용할 수 있습니다.

② 하드웨어 추상화 계층 (Hardware Abstraction Layer, HAL):

- 리눅스 커널과 하드웨어 간의 인터페이스 역할을 합니다.
- 카메라, 블루투스, GPS 등 다양한 하드웨어 장치에 대한 표준 인터페이스를 제공하여, 앱 개발자가 하드웨어의 복잡성을 직접 다루지 않고도 하드웨어 기능을 쉽게 사용할 수 있도록 돕습니다.

③ 안드로이드 런타임 (Android Runtime, ART) 및 네이티브 라이브러리:

- ART: 안드로이드 앱을 실행하는 가상 머신입니다. 자바나 코틀린으로 작성된 앱 코드를 기계어로 변환하여 실행하는 역할을 합니다.
- 네이티브 라이브러리: C/C++로 작성된 핵심 라이브러리들로, 그래픽, 데이터베이스, 웹 브라우징 등 시스템의 주요 기능을 담당합니다.

④ Java API 프레임워크:

- 안드로이드 앱을 개발할 때 가장 직접적으로 상호작용하는 계층입니다.
- 액티비티(Activity), 서비스(Service), 뷰(View) 시스템, 리소스 관리자 등 앱의 핵심 구성 요소를 쉽게 만들고 관리할 수 있는 API(Application Programming Interface)를 제공합니다.

⑤ 시스템 앱 (System Apps):

- 안드로이드 플랫폼에 기본적으로 포함된 앱들입니다.
- 전화, 메시지, 연락처, 웹 브라우저, 갤러리 등 사용자들이 스마트폰을 처음 켰을 때부터 사용하는 필수적인 앱들입니다.

결론적으로, '안드로이드 플랫폼'은 이 모든 계층(리눅스 커널, HAL, ART, 네이티브 라이브러리, Java API 프레임워크, 시스템 앱)을 아우르는 전체적인 소프트웨어 환경을 의미하며, 개발자들은 이 플랫폼이 제공하는 풍부한 기능과 도구를 활용하여 안드로이드 앱을 만들고 실행하게 됩니다.

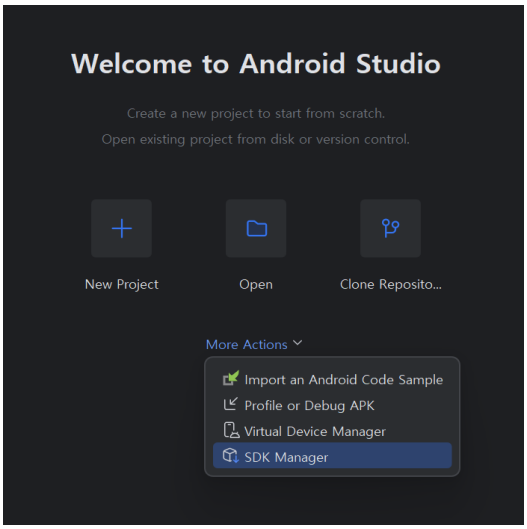


그림 1-8 SDK Manager

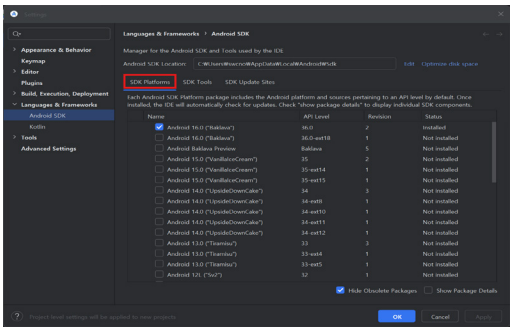


그림 1-9 SDK Platforms

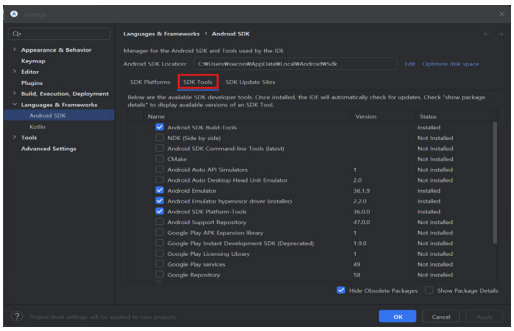


그림 1-10 SDK Tools

1.3.3 AVD(Android Virtual Device) 생성

이전 세션을 통해 Android Stduio 설치 및 환경설정과 Android 앱 개발환경을 마무리 하였습니다. 이번 단계는 개발된 Android 앱이 잘 동작하는지 여부를 확인하기 위한 테스트 환경설정 세션입니다. 테스트 환경은 물리적으로 크게 두가지 환경이 있을 수 있는데, 첫 번째는 가상환경(AVD)이고, 두 번째는 실제 안드로이드 기반 기기(phone, tablet, tv 등)입니다.

본 세션에서는 실제 안드로이드 기기 없이 가상 환경에서 앱을 테스트할 수 있도록 가상환경 기기를 생성하도록 설정할 것입니다.

표 1-2 AVD와 안드로이드 기반 기기 장단점

구분	AVD	안드로이드 기반 기기
장점	• 다양한 환경 테스트: 다양한 안드로이드 OS 버전, 화면 크기, 해상도, 하드웨어 사양을 가진 가상 기기를 손쉽게 생성할 수 있습니다. 이를 통해 여러 종류의 사용자 기기에서 앱이 어떻게 동작하는지 테스트할 수 있습니다. • 개발 편의성: USB 케이블 연결이 필요 없으며, 안드로이드 스튜디오 내에서 모든 테스트 환경을 제어할 수 있어 개발 워크플로우가 간편합니다. • 시뮬레이션 기능: GPS 위치, 전화, SMS, 배터리 상태, 네트워크 속도 등 실제 기기에서 일어날 수 있는 다양한 상황을 시뮬레이션할 수 있습니다. 이는 특정 상황에 대한 버그를 디버깅할 때 매우 유용합니다. • 스크린샷 및 동영상 녹화: 별도의 외부 도구 없이도 AVD의 화면을 손쉽게 캡처하거나 녹화할 수 있습니다. • 무료: 별도의 기기 구매 비용이 들지 않습니다.	• 최고의 성능: 앱이 실제 기기의 하드웨어 성능을 그대로 활용하므로, 에뮬레이터보다 훨씬 빠르고 부드러운 테스트가 가능합니다. 특히 게임이나 고성능 앱 테스트에 필수적입니다. • 정확한 동작 테스트: 센서(가속도계, 자이로스코프 등), 카메라, 블루투스, NFC와 같은 하드웨어 기능이 정확하게 동작하는지 테스트할 수 있습니다. • 현실적인 사용자 경험: 터치감, 진동 피드백, 배터리 소모량 등 실제 사용자가 경험하게 될 환경을 가장 정확하게 테스트할 수 있습니다. • 특정 버그 발견: 실제 기기에서만 발생하는 하드웨어 충돌, OS 커스터마이징 문제 등 에뮬레이터에서는 발견하기 어려운 버그를 찾을 수 있습니다.
단점	• 성능 문제: 에뮬레이터는 실제 하드웨어의 성능을 완벽하게 모방하지 못하며, 특히 그래픽 처리나 애니메이션에서 느리거나 끊기는 현상이 발생할 수 있습니다. • 실제 기기와 다른 동작: 가상 환경이므로, 하드웨어의 미묘한 차이(예: 센서의 정밀도)나 실제 기기에서만 발생하는 특정 버그를 발견하기 어려울 수 있습니다. • 초기 부팅 시간: 처음 AVD를 실행할 때 부팅 시간이 오래 걸릴 수 있습니다.	• 테스트 환경 제한: 보유하고 있는 기기에서만 테스트가 가능하므로, 다양한 안드로이드 버전이나 화면 크기에 대한 테스트를 하려면 여러 기기를 구매해야 합니다. • 준비 과정: 개발자 옵션 활성화, USB 디버깅 허용 등 초기 설정 과정이 필요하며, USB 케이블로 컴퓨터와 연결해야 합니다. • 기기 구매 비용: 다양한 기기에서 테스트하려면 기기를 여러 대 구매해야 하므로 비용이 발생합니다.

단점	• 시스템 리소스 소모: AVD는 컴퓨터의 CPU와 RAM을 많이 사용하므로, 저사양 컴퓨터에서는 여러 AVD를 동시에 실행하기 어렵거나 개발 환경 자체가 느려질 수 있습니다.	• 번거로움: USB 케이블 연결, 드라이버 설치 등 번거로운 과정이 있을 수 있으며, 특히 무선 디버깅 환경을 구축하지 않으면 물리적 연결이 필수적입니다.
추천	• 초기 개발단계와 호환성 테스트	• 마무리 단계와 최종 검증

• Device Manager 열기: [More Actions] 〉 Virtual Device Manager

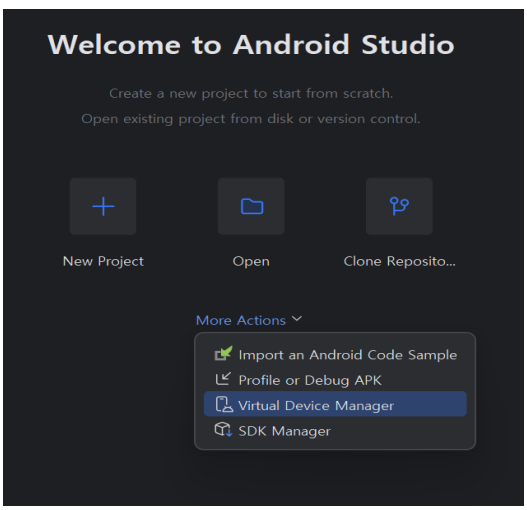
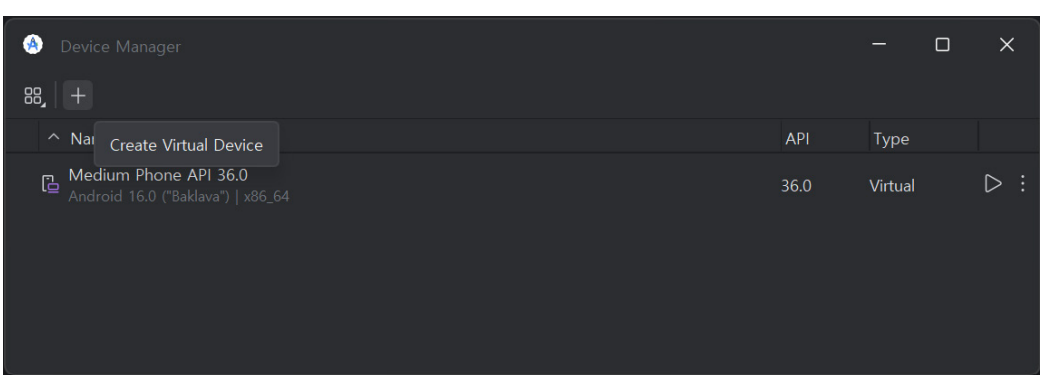


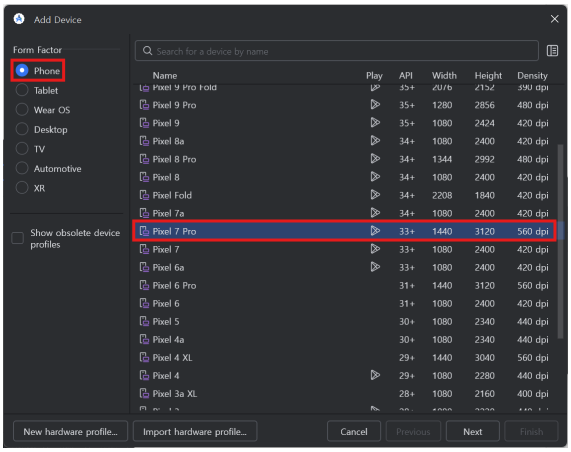
그림 1-11 Virtual Device Manager

• 가상 기기 생성: + "Create device" 버튼을 클릭합니다.



[Form Factor] 영역에서 테스트 기기 유형을 선택합니다. 여기서는 Phone을 선택합니다. 다음으로 원하는 하드웨어 프로파일을 선택합니다. (구글플레이 지원, 지원가능 API, 화면크기, 픽셀 등)

본 내용에서는 "Pixel 9 Pro"를 선택합니다.

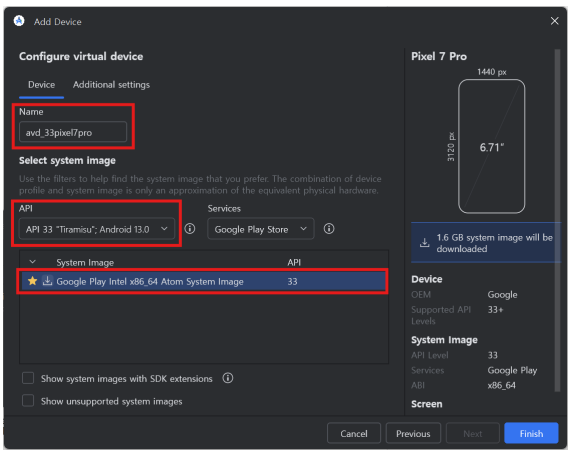


[Device] 〉 Name을 아래와 같이 변경한다. (avd_API 버전+프로파일명)

〉 API 변경을 변경한다. (해당 프로파일이 지원가능한 API)

〉 System Image 클릭

(클릭하지 않을 경우 [Finish]버튼 활성화 되지 않음)



아래 화면은 생성할 가상기기에 대한 세부 설정을 구성하는 단계입니다. 이번 세션에서는 각 영역별 간단한 소개만 하고, 앞으로 진행되는 과정에서 변경이 필요할 경우 해당 부분에서 자세히 다루기로 하겠습니다.

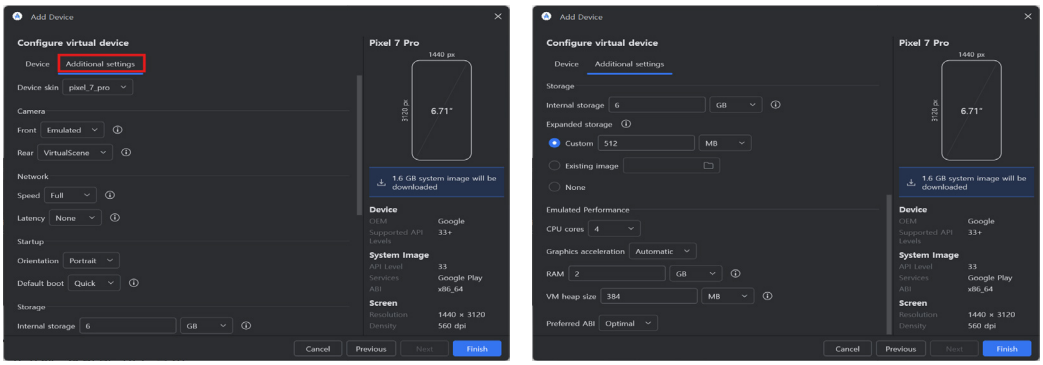


그림 1-12 AVD 기기생성 세부설정 화면

표 1-3 AVD 기기 생성 세부설정 항목별 내용

구분	설 명	
Device Skin	가상 기기의 외형(스킨)을 변경. 기본적으로 선택한 하드웨어 프로파일이 지정된다.	
Camera	전면 및 후면 카메라의 동작 방식을 설정합니다.	
	Front	'Emulated'(에뮬레이터에서 가상의 카메라를 사용) 또는 'Webcam'(실제 컴퓨터의 웹캠을 사용)을 선택할 수 있다.
	Rear	'VirtualScene'(가상환경) 또는 'Emulated'를 선택할 수 있다.
Network	Speed	네트워크 속도를 설정합니다. 'Full'(최고 속도), 'GSM', 'HSPA' 등 다양한 환경을 시뮬레이션할 수 있다.
	Latency	네트워크 지연 시간을 설정합니다. 'None' 또는 'GPRS', 'CDMA' 등 다양한 지연 환경을 시뮬레이션하여 네트워크 관련 버그를 테스트할 수 있습니다.
Startup	Orientation	가상 기기의 시작 방향을 'Portrait'(세로) 또는 'Landscape'(가로)로 설정한다.
	Default boot	'Quick boot'(빠른 부팅)을 설정하여 AVD 실행 속도를 단축할 수 있다.
Storage	Internal storage	가상 기기의 내부 저장 공간 용량을 설정합니다. 기본값으로 6GB가 설정되어 있으며, 필요에 따라 용량을 조절할 수 있다.
	Expanded storage	추가적인 저장 공간을 설정한다.

Emulated Performance	CPU cores	가상 기기가 사용할 CPU 코어의 수를 설정합니다. 코어 수가 많을수록 성능이 향상될 수 있지만, 컴퓨터의 CPU 리소스도 더 많이 사용된다.
	Graphics acceleration	그래픽 처리를 위한 가속 모드를 설정합니다. Automatic으로 설정하는 것이 일반적이며, 컴퓨터의 GPU를 활용하여 그래픽 렌더링 성능을 높힌다.
	RAM:	가상 기기가 사용할 RAM 용량을 설정합니다. 일반적으로 2GB가 기본값으로 설정되어 있다.
	VM heap size:	가상 머신 힙(Heap) 크기를 설정합니다. 앱이 할당받을 수 있는 최대 메모리 양에 영향을 준다.
	Preferred ABI:	가상 기기에서 사용할 명령어 세트(Application Binary Interface)를 선택합니다. 'Optimal'이 권장되며, 자동으로 최적의 ABI를 선택되게 된다.

모든 설정 완료 후 [Finish] 클릭합니다.

만약, 다운로드 되어 있지 않은 API 및 System Image를 선택했을 경우 자동으로 다운로드를 진행하게 되며, 본 세션에서도 API33 버전을 선택하였기에 다운로드를 진행하게 될 것입니다.

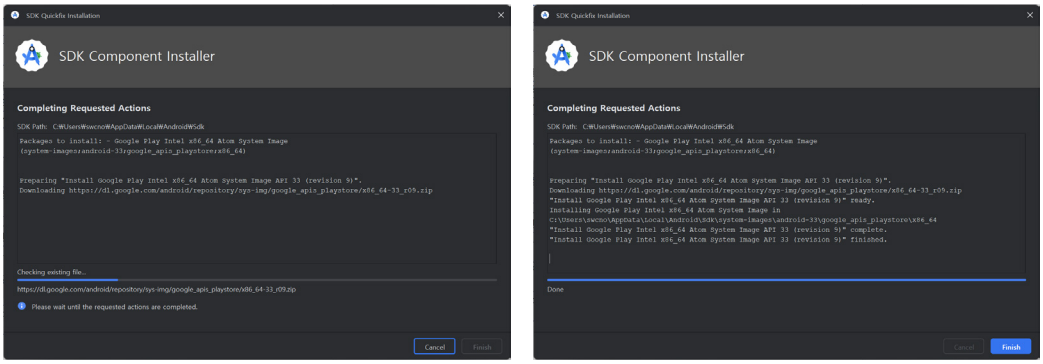


그림 1-13 SDK 다운로드 및 설치 화면

SDK 다운로드 및 설치가 완료되면 AVD생성 초기화면에 생성한 기기 목록이 표시되며, 생성된 기기를 선택하여 ▷ [Start] 버튼을 클릭하면 가상기기 화면이 컴퓨터 화면에 표시되게 됩니다.

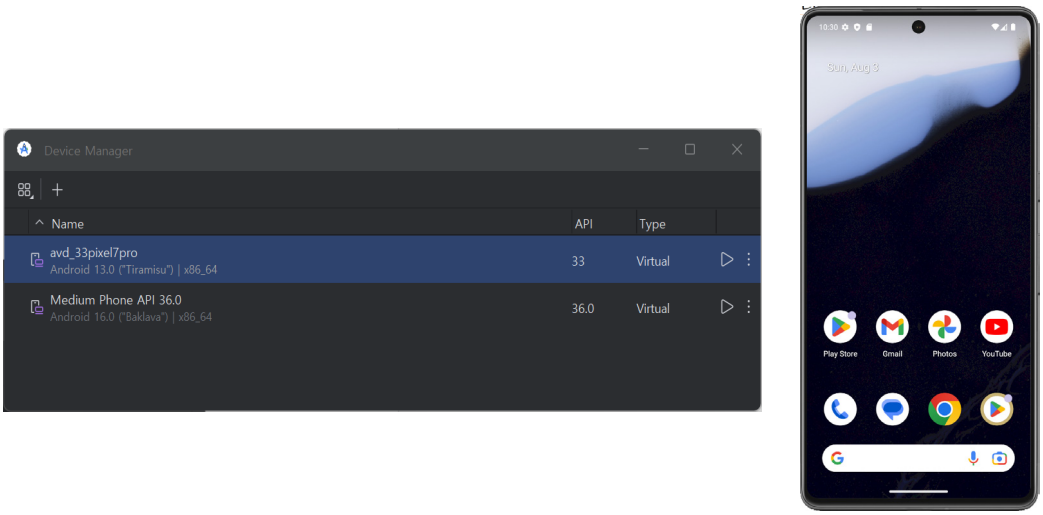


그림 1-14 AVD 가상기기 실행화면

연습문제

1. 안드로이드 앱의 특징에 대해 서술하세요.

2. 안드로이드 앱을 구성하는 4대 주요 컴포넌트(Activity, Service, Content, Provider, Broadcast Receiver) 각각의 역할을 설명하세요.

3. 안드로이드의 역사와 발전 과정을 간략히 서술하세요.

4. 안드로이드 개발환경을 구축하기 위해 필요한 필수 도구 3가지를 설명하고, 각각의 역할을 서술하세요.

5. AVD(가상 디비이스)와 실제 안드로이드 기기 테스트 환경의 장단점을 비교 설명 하세요.



제2장

안드로이드 앱의 기본 구조와 인터페이스



학습목표

- 안드로이드 앱의 기본 구조를 설명할 수 있다.
- View 계층과 레이아웃 시스템을 이해하고, Material 구성요소로 간단한 화면을 설계, 구현한다.

이 챕터에서는 안드로이드 앱이 어떤 요소들로 이루어져 있고, 이 요소들이 서로 어떻게 연결되어 하나의 완성된 애플리케이션을 구성하는지를 체계적으로 살펴봅니다. 또한, 사용자가 화면을 통해 앱과 직접 상호작용할 수 있도록 만드는 사용자 인터페이스(UI)를 설계하고 구현하는 기초적인 방법에 대해 학습하게 됩니다. 이러한 내용은 앱 개발을 처음 시작하는 단계에서 반드시 이해하고 익혀야 할 핵심 지식으로, 액티비티(Activity), 뷰(View), 레이아웃(Layout)과 같은 주요 구성 요소가 어떤 역할을 담당하는지, 그리고 이들이 데이터 처리 로직이나 이벤트 처리와 어떤 방식으로 연계되어 동작하는지를 파악하는 것이 중요합니다. 이를 통해 학습자는 단순히 화면을 만드는 것을 넘어, 앱의 구조를 전체적으로 이해하고 실무 개발에 필요한 기초 역량을 갖추게 될 것입니다.



안내

본 교재에서 사용되는 모든 미디어 (이미지)는 아래 URL 경로에서 다운 받을 수 있습니다. 아래 URL 경로에서 이미지를 특정경로에 다운로드 받아 교재에서 진행하는 프로젝트를 실습을 함께 진행하시면 됩니다.

- 미디어(이미지) 다운로드 경로(URL): https://buly.kr/FhO9yTI



또한, 본 교재에서 사용된 모든 코드도 아래 URL 경로에서 다운로드 받을 수 있습니다.

실습 진행시 오류나 작성이 어려운 코드는 아래 URL 경로에서 소스를 다운로드 받아 사용하시면 쉽게 문제를 해결할 수 있습니다.

- 실습 소스(Source) 다운로드 경로(URL): https://buly.kr/G3DI42e



2.1 안드로이드 앱 기본 구조

안드로이드 스튜디오에서 새로운 프로젝트를 생성하면, 앱 개발을 진행하는 데 필요한 여러 종류의 파일과 폴더들이 자동으로 생성되며, 이러한 구조는 단순히 형식적으로 존재하는 것이 아니라 앱의 전반적인 동작과 관리에 중요한 기반을 제공합니다. 예를 들어, 자바(Java) 또는 코틀린(Kotlin) 소스 코드가 저장되는 디렉터리, 사용자 인터페이스를 정의하는 XML 레이아웃 파일, 이미지나 문자열과 같은 리소스를 보관하는 폴더, 그리고 앱의 전반적인 설정과 권한을 지정하는 매니페스트 파일 등은 각각 고유한 역할을 담당하며 서로 긴밀하게 연결되어

있습니다. 이러한 프로젝트 구조를 이해하면 코드와 리소스를 체계적으로 배치할 수 있을 뿐만 아니라, 기능을 추가하거나 수정할 때 필요한 위치를 빠르게 찾을 수 있고, 유지보수와 확장 작업도 훨씬 효율적으로 수행할 수 있습니다. 따라서 초기에 프로젝트 구조를 정확히 이해하고 익히는 것은 원활한 앱 개발뿐만 아니라 향후의 관리와 품질 향상에도 필수적인 과정이라 할 수 있습니다.

2.1.1 프로젝트 뷰(Project View) 살펴보기

안드로이드 스튜디오에서 좌측에 위치한 Project 탭을 클릭하면, 현재 작업 중인 프로젝트에 포함된 모든 파일과 폴더의 구조를 한눈에 확인할 수 있습니다. 이 영역은 기본적으로 Android 뷰로 설정되어 있는데, 이는 안드로이드 애플리케이션 개발에 특화된 방식으로 각 파일과 폴더를 용도별로 분류하여 보여주는 형태입니다. 즉, 실제 물리적인 디렉터리 구조를 그대로 나열하기보다는, 개발자가 코드 작성과 리소스 관리, 설정 변경 등을 보다 직관적으로 수행할 수 있도록 논리적으로 재구성하여 제공합니다.

이 Android 뷰에서는 예를 들어, 소스 코드가 위치한 java 폴더, 사용자 인터페이스를 정의하는 res 폴더, 앱의 전반적인 정보와 권한을 지정하는 "AndroidManifest .xml" 파일 등이 명확하게 구분되어 표시됩니다.

또한 Gradle 스크립트나 프로젝트 설정 파일과 같이 빌드 및 환경 구성에 필요한 요소들도 별

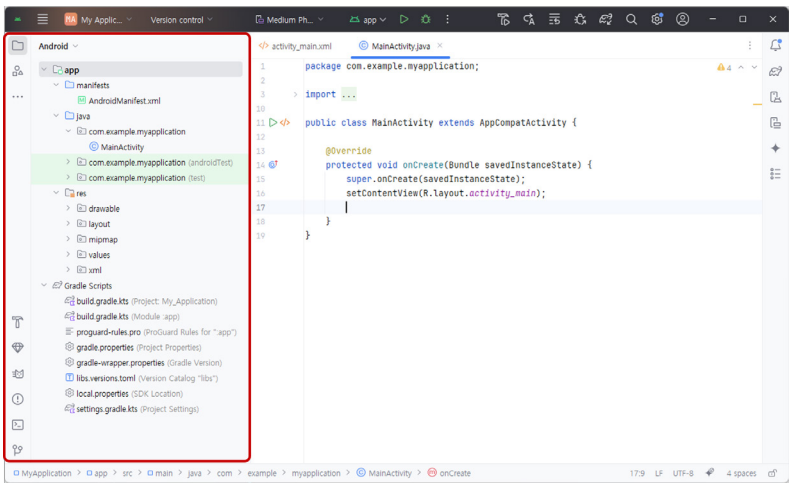


그림 2-1 Android Studio Project 뷰

도의 항목으로 정리되어 있어, 개발자가 프로젝트의 특정 부분을 빠르게 찾아 수정할 수 있습니다. 이러한 구조는 특히 초보 개발자에게 프로젝트의 구성 원리를 이해하는 데 큰 도움을 주며, 나아가 유지보수나 기능 확장 시에도 효율적인 작업 환경을 제공해 줍니다.

표 2-1 Android Project 파일 구조

구분	하위 폴더 및 파일 / 설명	
app	앱의 모든 소스 코드, 리소스, 매니페스트 파일 등이 포함된 주 모듈입니다. 대부분의 작업은 이 폴더 내에서 이루어집니다.	
manifest	AndroidManifest.xml 파일이 위치합니다.	
java (or Kotlin)	앱의 실제 로직을 담고 있는 소스 코드 파일(Java 또는 Kotlin)이 위치합니다.	
	com.example.yourappname	앱의 주요 액티비티, 클래스 파일 등이 이 안에 있습니다.
	com.example.yourappname (androidTest)	안드로이드 기기에서 실행되는 테스트 코드가 위치합니다.
	com.example.yourappname (test)	로컬 JVM에서 실행되는 단위 테스트 코드가 위치합니다.
res (resource)	앱에 사용되는 모든 리소스 파일(이미지, 레이아웃, 문자열, 색상 등)이 위치합니다.	
	drawable	이미지 파일 (PNG, JPG, XML 드로어블 등)
	layout	사용자 인터페이스 화면을 정의하는 XML 레이아웃 파일
	mipmap	앱 아이콘 파일 (다양한 해상도 지원)
	values	문자열(strings.xml), 색상(colors.xml), 스타일(styles.xml), 치수(dimens.xml) 등 앱 전반에 걸쳐 사용되는 값들을 정의하는 XML 파일
Gradle Scripts	Gradle 빌드 시스템 관련 파일들이 위치합니다.	

2.1.2 manifest -> AndroidManifest.xml

AndroidManifest.xml 파일은 안드로이드 애플리케이션의 전반적인 속성과 동작 방식을 정의하는 핵심 설정 파일이자, 일종의 "앱 설명서" 역할을 합니다. 이 파일은 Android Project 구조에서 app 폴더 하위의 manifest 디렉터리에 위치하며, 앱이 어떻게 동작해야 하는지에 대한 중요한 지침을 운영체제에 전달하는 역할을 수행합니다.

구체적으로 살펴보면, AndroidManifest.xml에는 앱의 이름, 아이콘, 버전 정보 등 기본적인 메타데이터가 포함되어 있어 사용자가 앱을 식별할 수 있도록 합니다. 뿐만 아니라, 앱을 구성하는 핵심 구성 요소인 액티비티(Activity), 서비스(Service), 콘텐츠 프로바이더(Content Provider),

브로드캐스트 리시버(Broadcast Receiver) 등의 존재와 속성, 그리고 이들이 어떻게 동작해야 하는지에 대한 정보도 명시됩니다. 또한 앱 실행 시 필요한 각종 권한(Permissions) — 예를 들어 인터넷 접속 권한, 카메라 사용 권한, 위치 정보 접근 권한 등 — 을 선언하여, 사용자가 앱 설치 또는 실행 전에 이러한 권한 요청을 확인하고 승인할 수 있도록 합니다.

결국, AndroidManifest.xml 은 안드로이드 운영체제가 해당 앱을 올바르게 인식하고 실행 할 수 있도록 도와주는 중추적인 안내서이며, 개발자가 앱의 구조와 동작원리를 설계하는데 있어서 반드시 이해하고 관리해야 하는 필수적인 파일이라 할수 있습니다. 다음은 AndroidManifest.xml 예시 파일입니다.

AndroidManifest.xml

```
<!-- AndroidManifest.xml 예시 -->
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- 다른 액티비티, 서비스 등을 추가하고 싶을 경우 추가 선언 가능 -->
    </application>

</manifest>
```

표 2-2 Androidmanifest.xml 주요 속성

구분	설명	
manifest	모든 안드로이드 매니페스트 파일의 루트 요소입니다. package 속성은 앱의 고유한 패키지 이름을 정의합니다.	
〈uses-permission〉	앱이 특정 기능을 수행하기 위해 필요한 권한을 선언합니다. 예를 들어, 〈uses-permission android:name="android.permission.INTERNET" /〉는 앱이 인터넷에 접근할 수 있는 권한이 필요함을 나타냅니다.	
〈application〉	앱의 전역 설정을 정의하는 요소입니다.	
	android:icon	앱의 아이콘을 지정합니다.
	android:label	앱의 이름을 지정합니다 (보통 strings.xml에 정의된 문자열 리소스를 참조).
	android:theme	앱 전체에 적용될 테마를 지정합니다.
〈activity〉	앱의 각 액티비티를 선언합니다.	
	android:name	액티비티 클래스의 이름을 지정합니다.
	android:exported ="true"	다른 앱에서 이 액티비티를 시작할 수 있도록 허용합니다. (Android 12 이상 필수)
	〈intent-filter〉	액티비티가 어떤 인텐트(Intent)에 응답할 수 있는지를 정의합니다. • 〈action android:name="android.intent.action.MAIN" /〉 → 해당 액티비티가 앱의 메인 액티비티임을 나타냄 • 〈category android:name="android.intent.category.LAU NCHER" /〉 → 해당 액티비티가 기기의 앱 런처에 표시되어야 함을 나타냄

2.1.3 Gradle Scripts

Gradle은 안드로이드 프로젝트의 빌드 시스템입니다. 앱을 컴파일하고, 필요한 라이브러리를 다운로드하며, 최종 APK(Android Package Kit) 파일을 생성하는 모든 과정을 Gradle이 담당합니다. 주요 Gradle 파일 구조는 다음과 같습니다.

표 2-3 Gradle 파일 구조

구분	설명
build.gradle (Project: YourAppName)	프로젝트 전체에 대한 설정 파일입니다. 주로 모든 모듈에 공통적으로 적용될 빌드 도구 버전 등을 정의합니다.
build.gradle (Module: app)	app 모듈에 대한 빌드 설정 파일입니다. 대부분의 앱별 설정(SDK 버전, 앱 버전, 의존성 라이브러리 등)이 여기에 정의됩니다.

아래는 build.grade (Module: app) 예시 파일입니다.

build.gradle (module.app)

```
// build.gradle (Module: app) 예시

plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "com.example.myapplication"
    compileSdk = 36  // 앱을 컴파일할 SDK 버전

    defaultConfig {
        applicationId = "com.example.myapplication"
        minSdk = 33          // 앱이 실행될 수 있는 최소 SDK 버전
        targetSdk = 36       // 앱이 대상으로 하는 SDK 버전 (최신 버전 권장)
        versionCode = 1      // 앱의 내부 버전 코드 (업데이트 시 증가)
        versionName = "1.0"  // 사용자에게 표시되는 앱 버전 이름

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}

dependencies {
    // 필요한 라이브러리 의존성 추가
    implementation(libs.appcompat)
    implementation(libs.material)
    implementation(libs.activity)
    implementation(libs.constraintlayout)
    testImplementation(libs.junit)
```

```
        androidTestImplementation(libs.ext.junit)
        androidTestImplementation(libs.espresso.core)
    }
```

표 2-4 build.gradle (Module: app) 파일의 주요 요소

요소	설명
compileSdk	앱을 컴파일하는 데 사용되는 안드로이드 SDK 버전을 지정합니다. 이 버전은 앱이 사용할 수 있는 API의 범위를 결정합니다.
minSdk	앱이 실행될 수 있는 최소 안드로이드 SDK 버전을 지정합니다. 이보다 낮은 버전의 기기에서는 앱이 설치되지 않습니다.
targetSdk	앱이 대상으로 하는 안드로이드 SDK 버전을 지정합니다. 이 버전은 앱이 최신 안드로이드 버전의 동작 변경 사항을 어떻게 처리할지 시스템에 알려줍니다.compileSdk와 동일한 버전으로 설정하는 것이 권장됩니다.
versionCode	앱의 내부 버전 번호로, 구글 플레이 스토어에 앱을 업데이트할 때마다 증가시켜야 합니다.
versionName	사용자에게 표시되는 앱의 버전 이름입니다 (예: "1.0", "1.1.2").
dependencies	앱이 사용하는 외부 라이브러리(예: androidx.appcompat, com.google.android. material)를 선언하는 부분입니다. Gradle은 여기에 선언된 라이브러리들을 자동으로 다운로드하여 프로젝트에 포함시킵니다.

2.2 인터페이스 기초

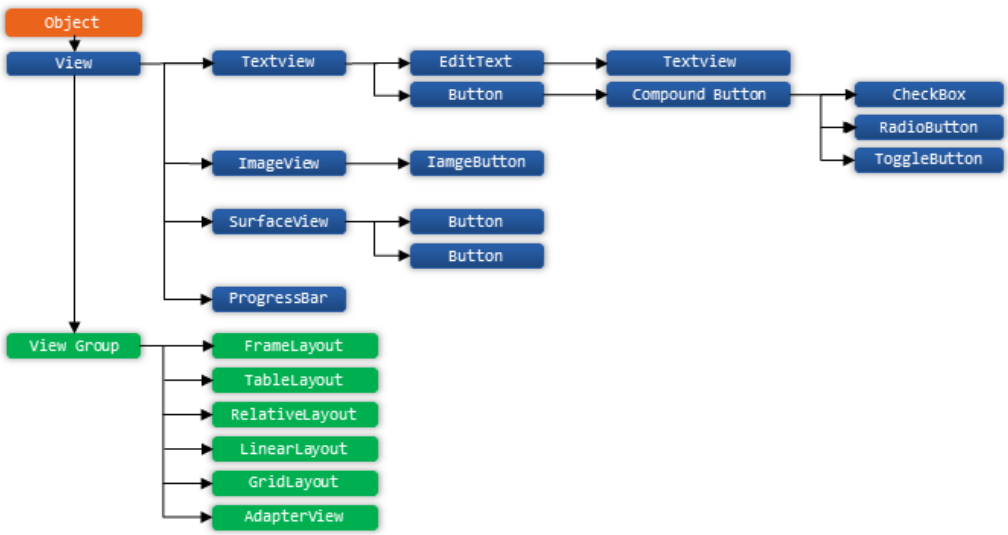


그림 2-2 Android View 클래스 구조

안드로이드 애플리케이션에서 사용자 인터페이스(UI)는 사용자가 앱과 직접 상호작용하는 시각적인 영역을 의미하며, 이는 앱의 사용성과 만족도를 결정짓는 중요한 요소입니다. 이러한 안드로이드 UI는 주로 XML 레이아웃 파일과 Java 또는 Kotlin 코드를 조합하여 구성됩니다. XML 레이아웃 파일은 화면의 구조와 배치를 선언적으로 정의하고, Java/Kotlin 코드는 그 화면에 동적인 동작과 로직을 부여하여 사용자 경험을 완성합니다.

안드로이드 기반의 스마트폰과 태블릿 기기는 제조사와 모델에 따라 매우 다양한 화면 크기와 해상도를 제공하므로, 동일한 앱이라도 기기 환경에 맞게 화면이 올바르게 표시되도록 레이아웃을 유연하게 조정하는 것이 필수적입니다. 이를 위해 안드로이드 개발에서는 크기 단위(dp, sp)와 반응형 레이아웃 설계 방식을 활용하여 다양한 화면 환경에서도 일관된 사용자 경험을 유지합니다.

또한, [그림 2-2]에서 확인할 수 있듯이, 안드로이드 UI의 구성 요소는 크게 두 가지 계층으로 나눌 수 있습니다. 첫 번째는 View 클래스와 그 하위(sub) 클래스에 해당하는 요소들로, 이를 일반적으로 뷰(View) 또는 **위젯(Widget)**이라고 부릅니다. 예를 들어 버튼(Button), 텍스트뷰(TextView), 이미지뷰(ImageView) 등이 여기에 해당합니다. 두 번째는 ViewGroup 클래스와 그 하위(sub) 클래스에 속하는 요소들로, 이러한 구성 요소들은 **레이아웃(Layout)**이라 불리며, 여러 개의 뷰나 다른 레이아웃을 포함하여 화면의 배치를 관리합니다. 대표적인 예로 LinearLayout, RelativeLayout, ConstraintLayout 등이 있습니다.

결국, 안드로이드 UI 설계는 뷰와 레이아웃의 올바른 조합과 배치를 통해, 다양한 화면 크기와 해상도에서도 사용자가 편리하게 앱을 이용할 수 있도록 만드는 과정이라 할 수 있습니다.

2.2.1 액티비티(Activity)와 레이아웃(Layout)

- 액티비티(Activity) : 앞서 설명했듯이, 액티비티는 사용자 인터페이스를 담당하는 하나의 화면 단위입니다. 앱의 각 화면은 하나의 액티비티에 대응된다고 생각할 수 있습니다.
- 레이아웃(Layout): 액티비티 화면에 표시될 UI 요소(버튼, 텍스트, 이미지 등)를 어떻게 배치할지 정의하는 틀입니다. 레이아웃은 XML 파일로 작성되며, res/layout 폴더에 위치합니다.

안드로이드 Layout에는 그림 2-2에서 보는 바와 같이 LinearLayout, Relative Layout, TableLayout, FrameLayout, AbsoluteLayout, GridLayout 외에도 Constraint Layout, CoordinatorLayout 이 있다. 이러한 Layout 중에는 단독으로 사용할 수도 있지만, 하나의

Layout 안에 또 다른 Layout을 포함 시킬수 있다.

아래 그림 2-3과 같이 Root View 요소인 RelativeLayout 아래 Linearlayout, TableLayout이 포함되어 있는 것을 확인할 수 있을 것이다.

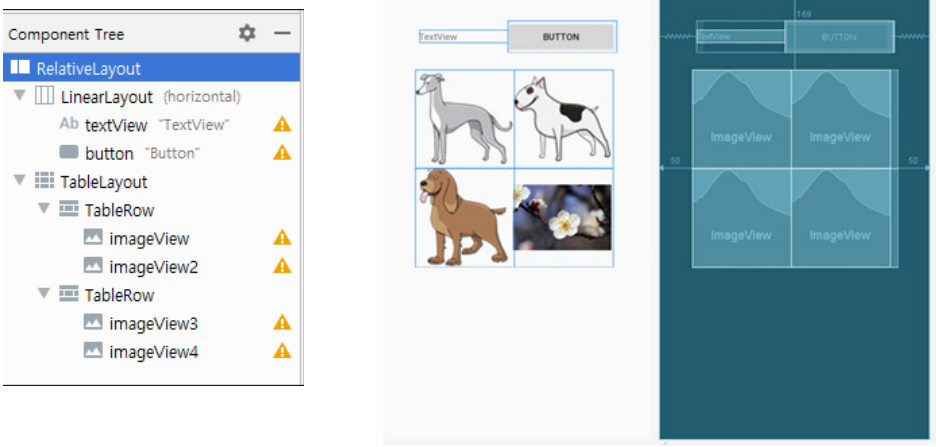


그림 2-3 하나의 Layout 안에 중첩Layout 구성

앞으로 자주 다룰 예정이지만, 2장에서 가장 일반적으로 사용되는 레이아웃을 먼저 소개하고 점차적으로 자세히 다루도록 하겠습니다.

① LinearLayout: 자식 뷰들을 수직(vertical) 또는 수평(horizontal)으로 일렬로 배치합니다.

activity_main.xml

```
    <!-- LinearLayout 예시 -->

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/main"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />

</LinearLayout>
```

② RelativeLayout: 자식 뷰들을 다른 뷰나 부모 레이아웃에 상대적으로 배치합니다. (예: "이 버튼은 저 텍스트 뷰 아래에", "이 이미지는 부모의 중앙에")

activity_main.xml

```
    <!-- RelativeLayout 예시 -->

    <RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <TextView
            android:id="@+id/myTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="중앙 텍스트" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/myTextView"
            android:layout_centerHorizontal="true"
            android:text="아래 버튼" />
    </RelativeLayout>
```

③ ConstraintLayout: 가장 유연하고 강력한 레이아웃으로, 각 뷰에 제약 조건(constraint)을 부여하여 복잡한 UI를 효율적으로 구성할 수 있습니다. (최근 안드로이드 개발에서 가장 권장되는 레이아웃)

activity_main.xml

```
<!-- ConstraintLayout 예시 -->

<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello ConstraintLayout!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/myTextView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2.2.2 뷰(View)와 위젯(Widget)

안드로이드 뷰(View)클래스 내 모든 뷰(View)는 화면에 보이는 모든 UI 요소의 가장 기본적인 빌딩 블록입니다. 텍스트, 이미지, 버튼 등 사용자가 보고 상호작용할 수 있는 모든 것이 뷰입니다. 위젯(Widget)은 특정 기능을 수행하는 미리 정의된 뷰의 한 종류이며, component 라고도 말한다.

자주 사용되는 뷰/위젯은 다음과 같습니다.

- TextView: 텍스트를 표시하는 뷰입니다.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="이것은 텍스트입니다."
    android:textSize="20sp"
    android:textColor="#FF0000" />
```

- Button: 클릭 가능한 버튼입니다.

```
<EditText
    android:id="@+id/myEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="여기에 입력하세요"
    android:inputType="text" />
```

- EditText: 사용자로부터 텍스트 입력을 받는 뷰입니다.

```
<EditText
    android:id="@+id/myEditText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="여기에 입력하세요"
    android:inputType="text" />
```

- ImageView: 이미지를 표시하는 뷰입니다.

```
<ImageView
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:src="@drawable/my_image" /> <!-- res/drawable 폴더의 my_image.png 참조 -->
```

위에 언급된 뷰/위젯을 포함한 모든 위젯은 공통적으로 다음과 같은 중요한 속성을 가집니다.

표 2-5 뷰/위젯 공통 속성

속성	설명
android:id	뷰를 고유하게 식별하는 ID입니다. 코드에서 이 ID를 사용하여 뷰에 접근합니다. @+id/는 새로운 ID를 정의할 때 사용됩니다.
android:layout_width	뷰의 너비를 지정합니다. • match_parent: 부모의 뷰 크기만큼 채움 • wrap_content: 뷰의 내용물 크기만큼만 차지합니다.
android:layout_height	뷰의 높이를 지정합니다. • match_parent: 부모의 뷰 크기만큼 채움 • wrap_content: 뷰의 내용물 크기만큼만 차지합니다.
android:text	TextView나 Button에 표시될 텍스트를 지정합니다.
android:hint	EditText에 입력 가이드로 표시될 텍스트를 지정합니다.
android:src	ImageView에 표시될 이미지 리소스를 지정합니다.

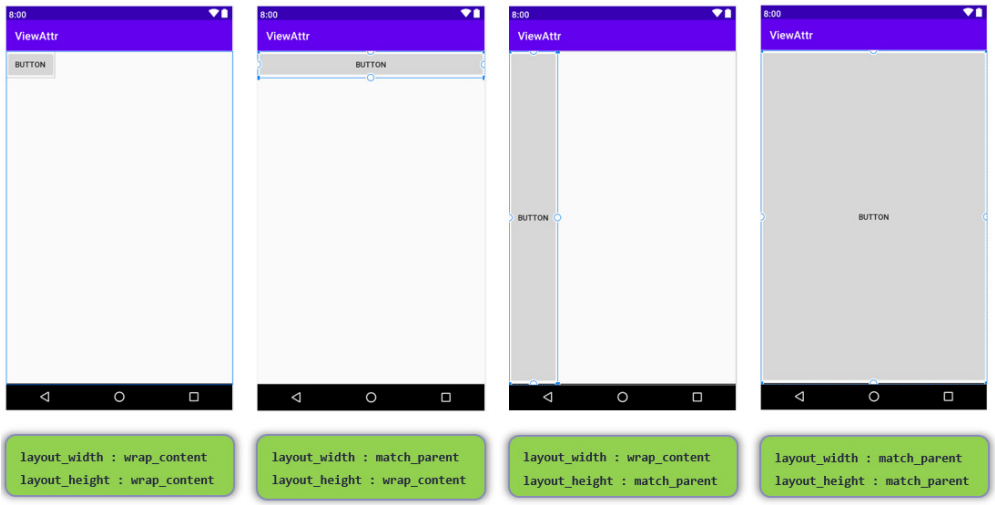


그림 2-4 layout_width, layout_height 요소 값에 따른 화면배치

2.2.3 XML 레이아웃과 코드 연결하기

XML 레이아웃 파일에 정의된 뷰들은 액티비티의 Java/Kotlin 코드에서 접근하고 제어할 수 있습니다. 이를 위해 뷰에 부여된 id를 사용합니다.

MainActivity.java

```
// MainActivity.java 예시

package com.example.myapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.TextView;
import android.view.View; // View 클래스 임포트
public class MainActivity extends AppCompatActivity {
    // 뷰 객체를 참조할 변수 선언
    private TextView myTextView;
    private Button myButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // 액티비티가 사용할 레이아웃 파일을 설정합니다.
        setContentView(R.layout.activity_main);
        // XML 레이아웃의 뷰를 ID를 통해 찾아와 변수에 연결합니다.
        myTextView = findViewById(R.id.myTextView);
        myButton = findViewById(R.id.myButton);
        // TextView의 텍스트 변경
        myTextView.setText("버튼을 눌러보세요!");
        // 버튼 클릭 이벤트 리스너 설정
        myButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 버튼이 클릭되었을 때 실행될 코드
                myTextView.setText("버튼이 클릭되었습니다!");
            }
        });
    }
```



코드 설명

- setContentView(R.layout.activity_main);
  onCreate() 메서드에서 이 액티비티가 사용할 XML 레이아웃 파일(activity_main.xml)을 지정합니다. R.layout.activity_main은 res/layout 폴더에 있는 activity_main 레이아웃 파일을참조하는 안드로이드의 리소스 ID입니다.

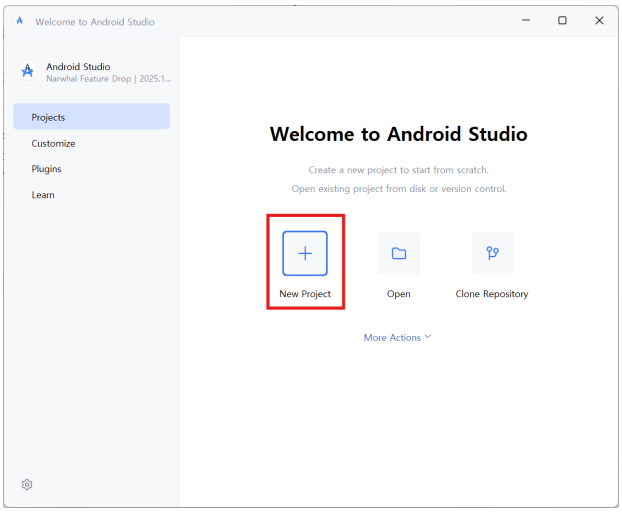
- findViewById(R.id.myTextView); :
  XML 레이아웃 파일에서 android:id="@+id/myTextView"로 정의된 TextView를 찾아와 Java/ Kotlin 코드의 myTextView 변수에 연결합니다.
  R.id.myTextView는 res/values/ids.xml에 자동으로 생성되는 ID를 참조합니다.
- myTextView.setText("..."); : 연결된 TextView 객체의 setText() 메서드를 사용하여 화면에 표시되는 텍스트를 변경합니다.
- myButton.setOnClickListener(...): Button 객체에 클릭 이벤트 리스너를 설정합니다. 사용자가 버튼을 클릭하면 onClick() 메서드 안의 코드가 실행됩니다.

2.3 Layout 실습

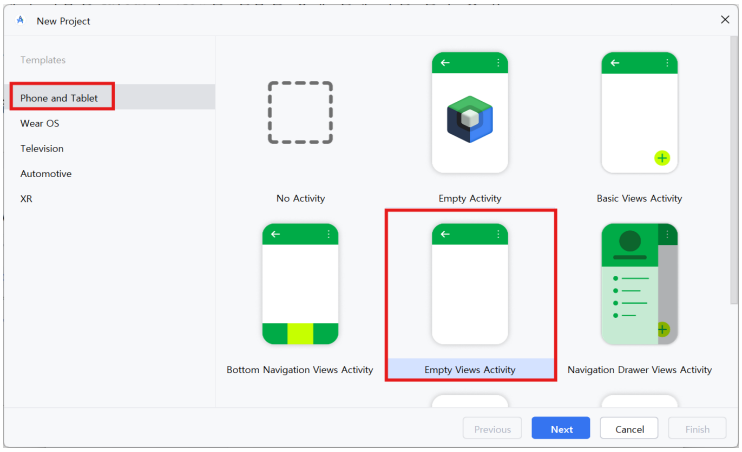
이번 세션에서는 앞에서 살펴 보았던 레이아웃(Layout)과 뷰(View)의 개념을 바탕으로, 실제 실습을 통해 이들이 화면에서 어떻게 표현되고 변화하는지를 직접 확인해 보겠습니다. 단순히 이론적으로 구조와 속성을 이해하는 데 그치지 않고, 다양한 속성 값을 변경하거나 배치를 조정함으로써 화면 구성 요소들이 어떤 방식으로 반응하고, 그 결과로 전체 UI가 어떻게 변하는지를 체험하게 됩니다. 이를 통해 각 뷰와 위젯이 갖는 특성과 역할, 그리고 레이아웃이 화면 요소를 배치하고 정렬하는 원리를 보다 직관적으로 이해할 수 있습니다. 궁극적으로는 이러한 실습 과정을 통해 UI 구성에 대한 이해도를 한층 높이고, 향후 실제 앱 개발 과정에서 보다 효율적이고 사용자 친화적인 화면을 설계할 수 있는 역량을 강화하는 것을 목표로 합니다.

2.3.1 LinearLayout 실습

Android Studio Welcome 화면에서 [New Project]를 선택한다.

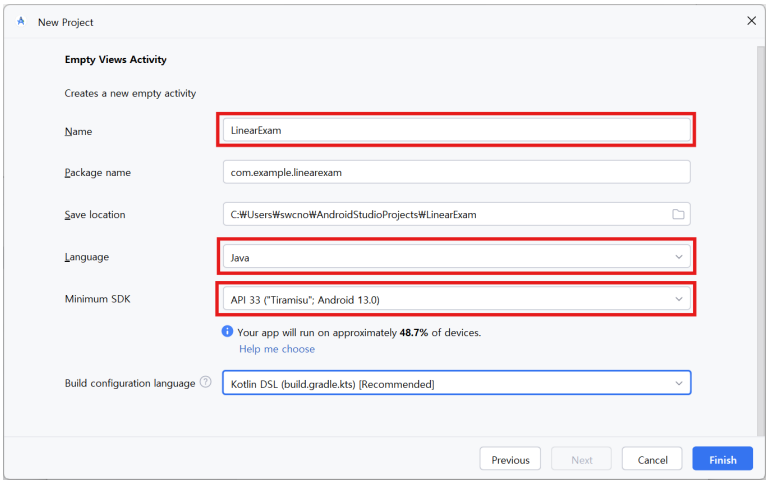


다음 화면에서, 개발 기기는 [Phone and Tablet]을 선택하고 [Empty Views Activity]를 선택한다.

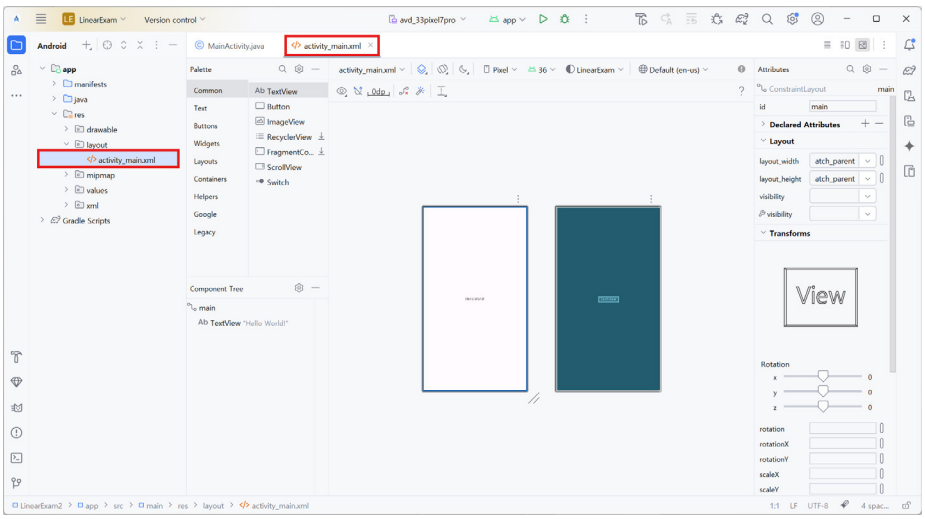


다음 화면에서, 아래 항목을 아래 그림처럼 변경하세요.

- Name : LinearExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

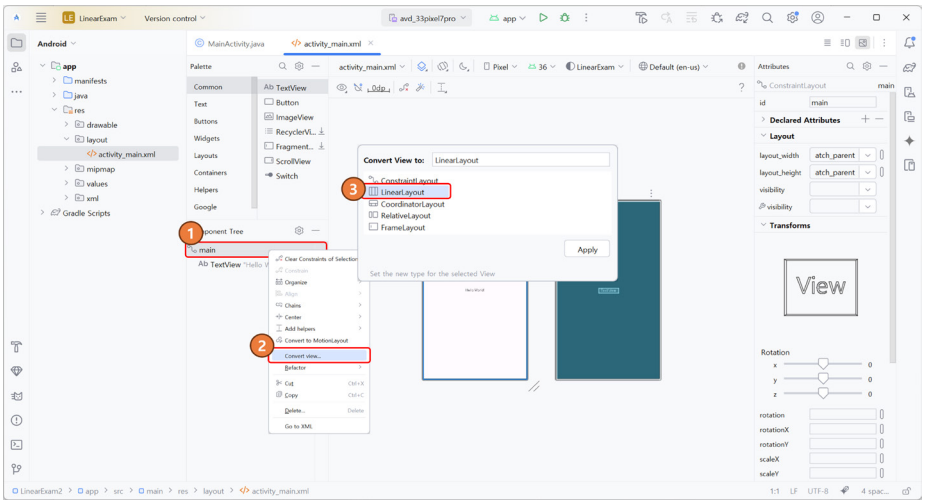


아래 화면과 같이 Project Navigator에서 [app]–[res]–[activity.xml]을 더블 클릭한다.



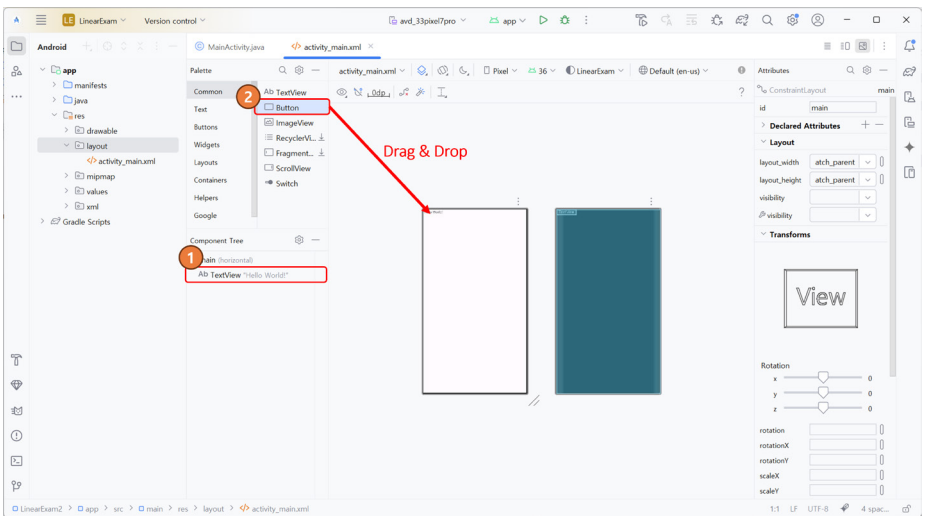
다음 순서로 layout을 변경하세요.

- ❶ 마우스 우측 클릭
- ❷ [Convert view...] 클릭
- ❸ [LinearLayout] 클릭

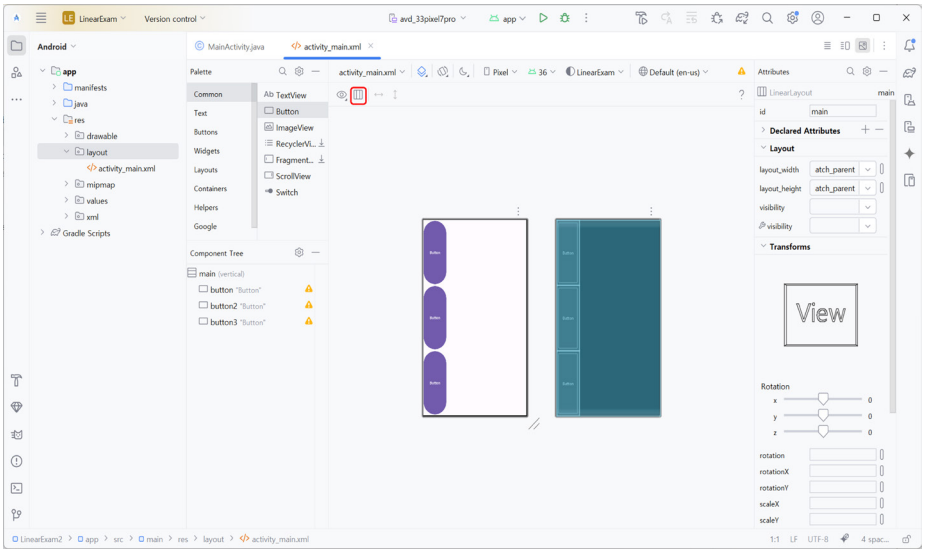


아래 그림과 같이 [TextView]를 선택한 후 키보드 [Delete] 키로 삭제하고,

[Palette]에서 [Button] widget을 아래 화면과 같이 〈〈Drag & Drop〉〉 3번 연속 진행 하세요.



아래 그림의 붉은 색 표시 된 부분(convert orientation to vertical) 을 클릭하여 화면 배치가 가로(horizontal)에서 세로(vertical)로 변경되는지 확인해 보세요.



2.3.2 RelativeLayout 실습

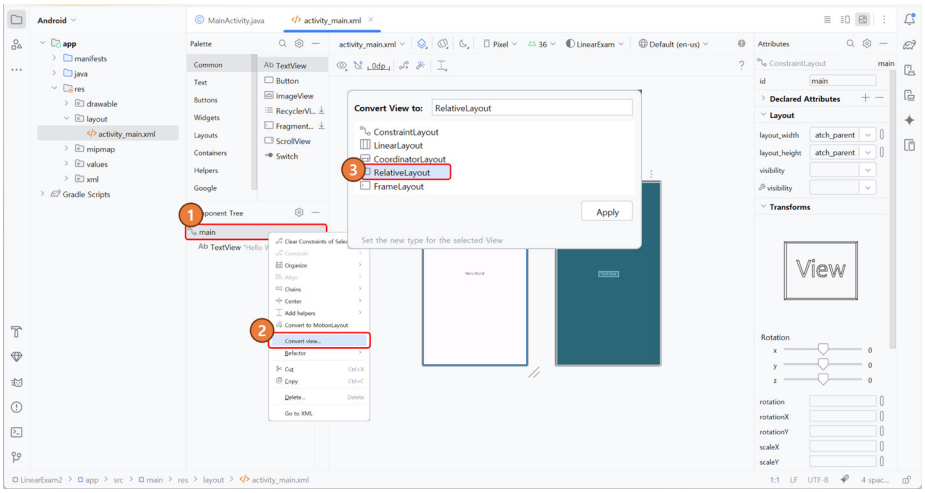
이번에 진행할 실습에서는 RelativeLayout을 활용하게 되는데, 이는 이전 세션에서 설명한 것처럼 각 뷰(View)와 위젯(Widget) 간의 상대적인(Relative) 위치 관계를 기반으로 화면 요소를 배치하는 레이아웃 방식입니다. 이러한 특성 덕분에, 단순히 위에서 아래로 또는 왼쪽에서 오른쪽으로만 배치하는 방식이 아니라, 다른 요소의 위치를 기준으로 유동적이고 유연하게 화면을 구성할 수 있다는 장점이 있습니다. 예를 들어, 특정 버튼을 화면 중앙에 배치하거나, 텍스트를 이미지의 바로 아래에 정렬하는 등 다양한 레이아웃 구성이 가능합니다.

이번 실습에서는 앞서 진행했던 "2.3.1 LinearLayout" 실습 내용을 참고하여, 동일한 절차와 기본 설정 방법을 바탕으로 새로운 프로젝트를 생성하고, 그 안에서 RelativeLayout을 적용한 화면을 구현해 보겠습니다. 이를 통해 LinearLayout과 RelativeLayout의 차이점과 장단점을 비교하며, 상황에 따라 적합한 레이아웃을 선택하고 활용하는 능력을 기를 수 있도록 하겠습니다.

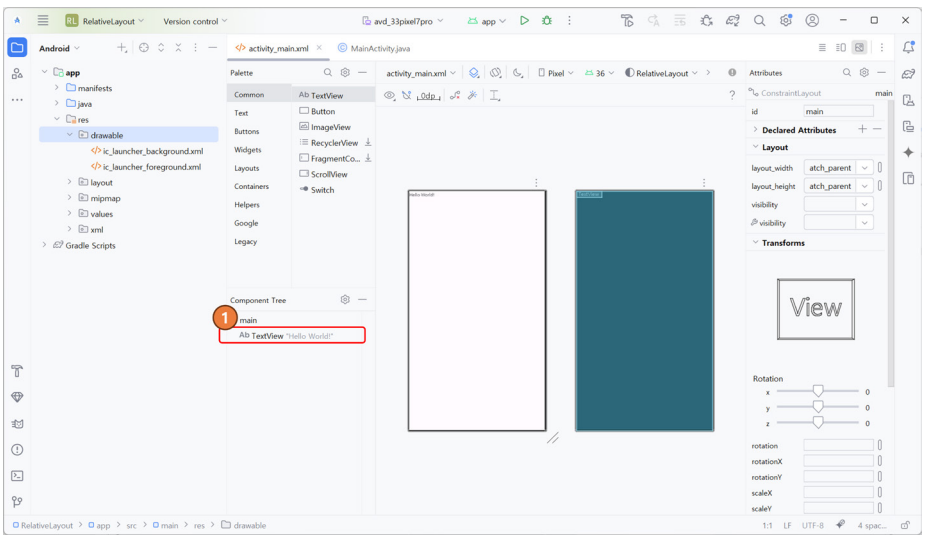
- Name : RelativeExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

다음 링크를 클릭하여 Project에 사용할 이미지를 다운로드 하세요.

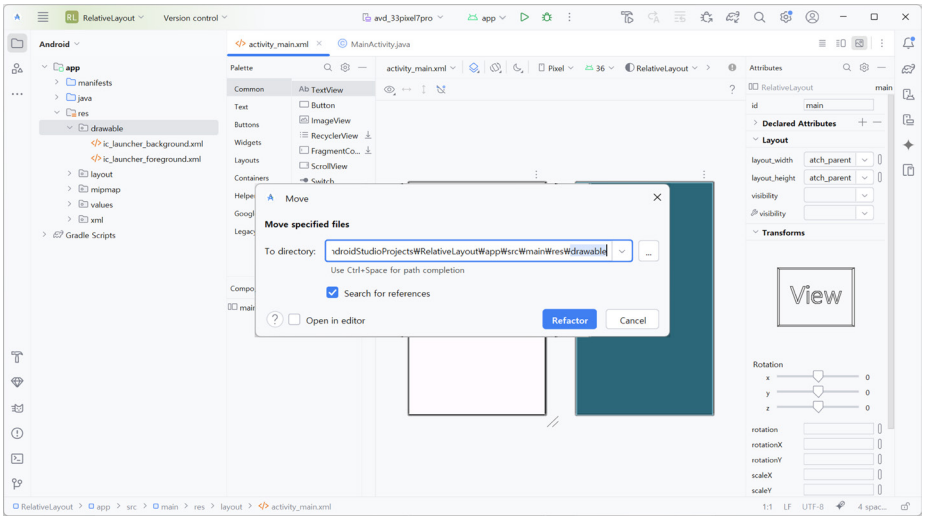
- 다운로드 url : https://buly.kr/FhO9yTl

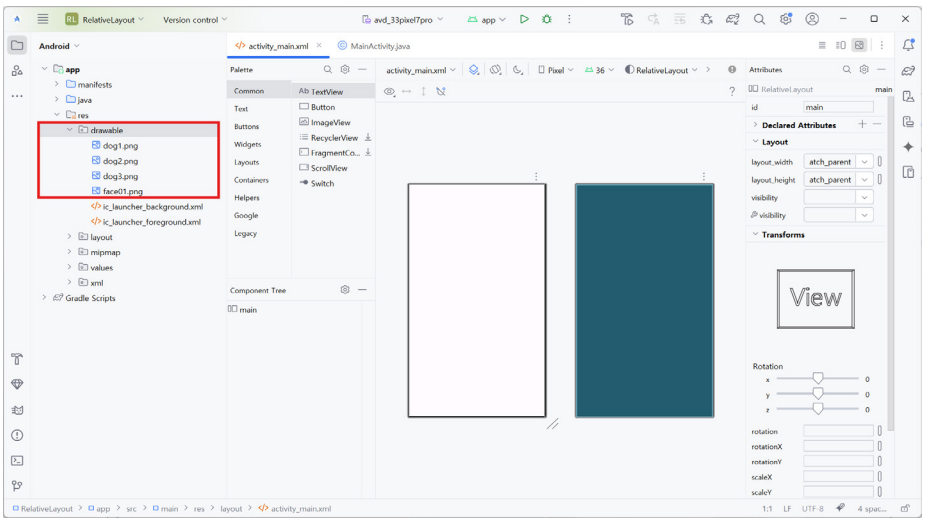


아래 화면에 표시된 [TextView]를 삭제하세요.



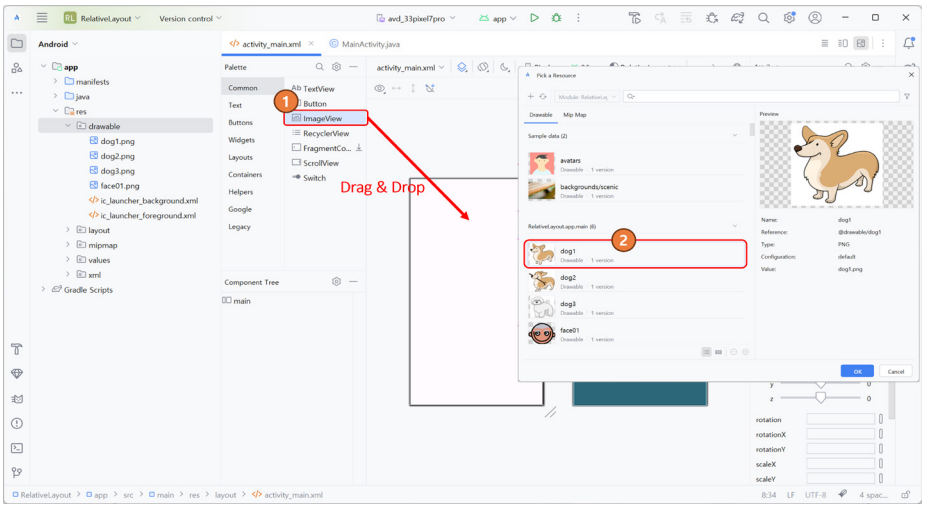
아래 화면과 같이 다운로드 한 이미지 중 dog1, dog2, dog3, face01 이미지를 [res]–[drawable]에 drag & drop 하세요.



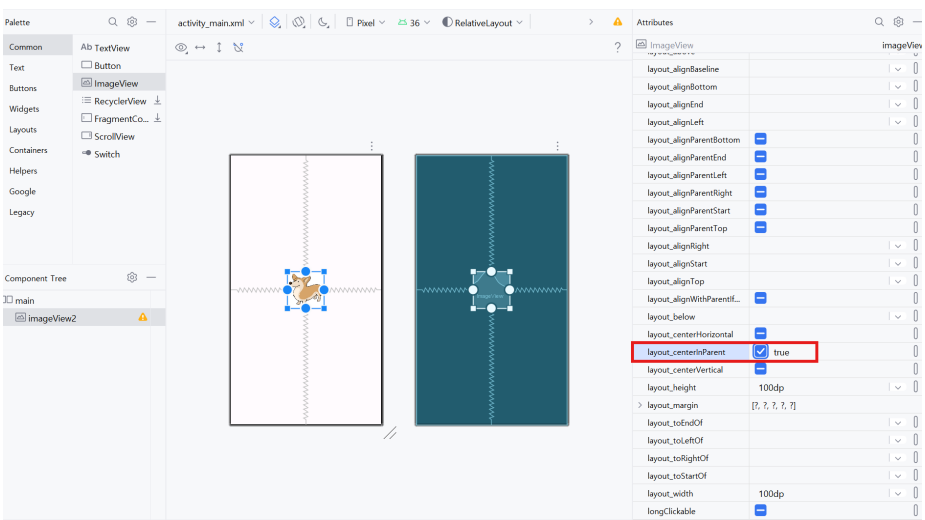


다음 과정으로 [ImageView]를 선택 후 drag & drop 하고, dog1 이미지를 선택하세요.

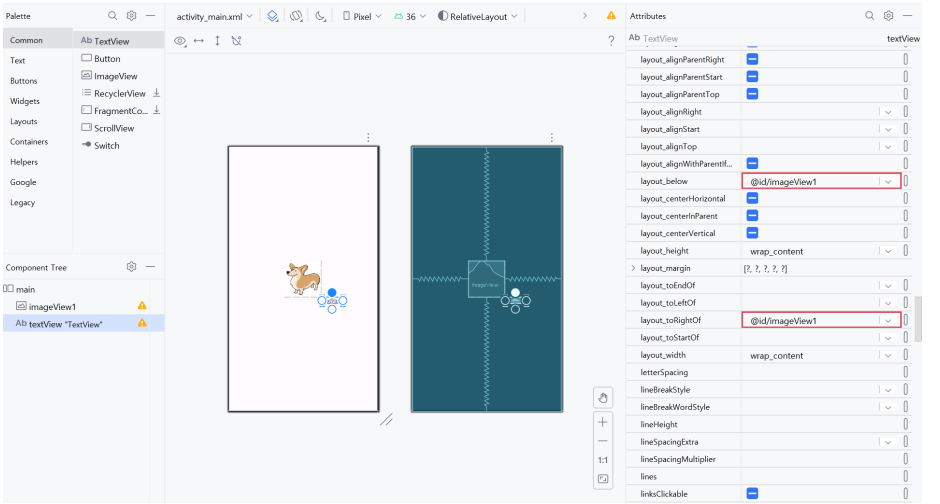
[ImageView]의 속성 중 layout_align 으로 시작된 속성들 값을 초기화 시켜주세요.



layout_width, layout_height 는 각 100dp로 설정하고, layout_centerInParent를 클릭하여 true로 변경하면, 아래 그림처럼 이미지가 화면 중앙에 배치 되어집니다.



다음으로 Palette에서 [TextView]를 선택하여 drag & drop 후, [TextView] 속성 중 layout_below(하단)와 layout_toRightOf(오른쪽) 값을 @id/imageView1로 설정하세요.

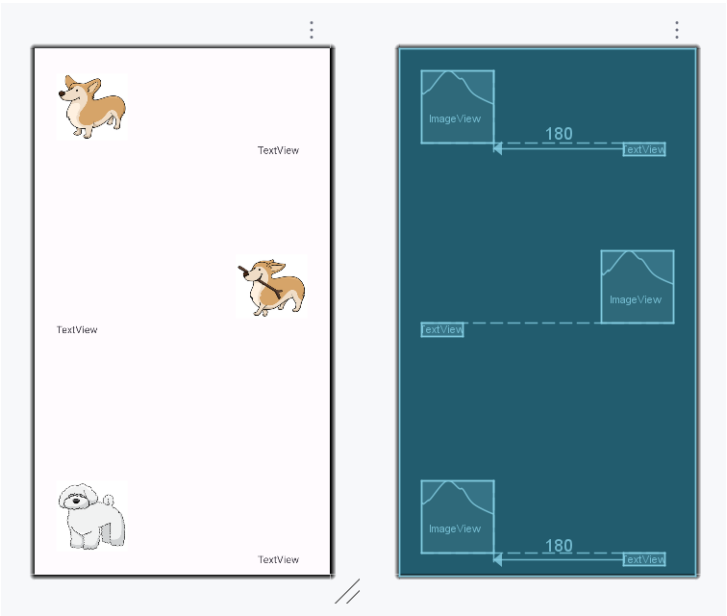


위와 같은 방법으로 아래와 같이 뷰/위젯을 추가하고, 변경해 보세요.

뷰/위젯 id	속성값	
imageView1	layout_width	100dp
	layout_height	100dp
	layout_marginStart	30dp
	layout_marginTop	30dp
imageView2	layout_width	100dp
	layout_height	100dp
	layout_marginLeft	280dp
	layout_marginTop	280dp
imageView3	layout_width	100dp
	layout_height	100dp
	layout_marginStart	30dp
	layout_marginTop	600dp
textView1	layout_width	wrap_content
	layout_height	wrap_content
	layout_below	@id/imageView1
	layout_marginLeft	180dp

textView2	layout_toRightOf	@id/imageView1
	layout_width	wrap_content
	layout_height	wrap_content
	layout_below	@id/imageView2
textView3	layout_marginTop	30dp
	layout_width	wrap_content
	layout_height	wrap_content
	layout_below	@id/imageView3
	layout_marginLeft	180dp
	layout_toRightOf	@id/imageView3

뷰/위젯 설정을 완료하면 아래와 같이 화면 구성이 완료 됩니다. 확인해 보세요.



연습문제

1. 안드로이드스튜디오에서 새로운 프로젝트를 생성했을 때 자동으로 생성되는 주요파일과 폴더에는 어떤 것들이 있으며, 각각 어떤 역할을 담당하는지 설명하세요.

2. 액티비티(Activity)와 뷰(View)는 안드로이드 앱에서 어떤 역할을 수행하며, 두 요소가 어떻게 연결되어 사용자 인터페이스를 구성하는지 기술하세요.

3. 안드로이드 앱에서 레이아웃(Layout)의 의미와 역할을 설명하고, LinearLayout과 RelativeLayout의 차이를 예를 들어 설명하세요.

4. 안드로이드에서 제공하는 Material Design 구성 요소(Material Components)의 특징과 이를 활용하여 사용자 인터페이스(UI)를 구현할 때의 장점을 서술하세요.

5. 매니페스트 파일이 앱 실행과 관리에 있어 가지는 중요성을 설명하고, 대표적으로 어떤 정보(예: 권한, 구성요소 선언 등)를 포함하는지 기술하세요.



제3장

이벤트(Event)



학습목표

- 안드로이드에서 Event 처리 방식을 이해한다.
- 익명클래스, 참조객체 방식 등 각 이벤트 처리 방식에 대해 고찰하고, 실제 작성할 수 있다.

안드로이드 앱은 기본적으로 사용자와의 상호작용을 통해 동작합니다. 사용자가 화면을 터치하거나 버튼을 클릭하고, 텍스트를 입력하는 등의 모든 행위는 이벤트(Event)를 발생시키며, 앱은 이러한 이벤트를 감지하고 그에 맞게 반응함으로써 동적인 사용자 경험을 제공합니다. 이 챕터에서는 안드로이드 앱에서 발생하는 다양한 이벤트를 어떻게 처리하는지, 그리고 이를 구현하는 여러 방법을 심층적으로 학습하게 됩니다.

여기서 말하는 이벤트란, 스마트폰 화면에서 사용자가 행하는 모든 동작을 포괄적으로 의미합니다. 예를 들어 화면을 한 번 터치하는 동작, 연속으로 터치하는 동작, 특정 뷰(View)를 터치하는 동작, 길게 누르는 동작 등은 모두 이벤트에 해당합니다. 이론적으로는 개발자가 이러한 모든 동작에 대응하는 코드를 작성할 수도 있지만, 실제 개발에서는 앱의 기능과 목적에 맞는 특정 이벤트에 대해서만 코드를 작성하여 처리합니다. 이렇게 함으로써 불필요한 기능을 제한하고, 사용자가 앱을 예측 가능한 방식으로 사용할 수 있도록 제약을 두게 됩니다.

예를 들어, 화면에 버튼이 하나 있다고 가정해 봅시다. 만약 개발자가 그 버튼의 '클릭(Click)' 동작에 대해서만 이벤트 처리 코드를 작성하고, 나머지 동작(길게 누르기, 더블 클릭 등)에 대해서는 아무런 코드를 작성하지 않았다면, 사용자는 그 버튼을 클릭했을 때만 동작을 수행할 수 있습니다. 반대로 버튼을 길게 누르거나 연속 클릭하는 등의 동작을 취하더라도, 앱은 아무런 반응을 하지 않게 됩니다.

즉, 이벤트 처리는 앱이 사용자와 상호작용하는 핵심 메커니즘이며, 어떤 이벤트를 처리할지 선택하는 것은 앱의 사용성과 사용자 경험을 좌우하는 중요한 설계 과정입니다.
이러한 Event 를 처리하도록 코드를 작성하는 방법에는 다양한 방법이 있습니다.

- 익명 클래스
- 익명 클래스로 참조 객체 생성
- 인터페이스 상속
- component(widget)의 onClick 속성 값 지정

이 방법들 중 개인의 취향 및 상황에 맞게 적절한 방법을 택하여 구현하면 됩니다.

3.1 익명 클래스(Anonymous Class)

안드로이드에서 이벤트를 처리하는 가장 일반적인 방법 중 하나는 익명 클래스(Anonymous Class)를 활용하는 것입니다. 익명 클래스는 말 그대로 이름이 없는 클래스이며, 주로 특정 인터페

이스나 추상 클래스를 단 한 번만 구현하여 사용할 경우에 유용하게 쓰입니다. 이를 통해 별도의 클래스를 정의하지 않고도, 필요한 위치에서 즉시 이벤트 처리 로직을 작성할 수 있어 코드의 간결성과 가독성을 높일 수 있습니다.

3.1.1 익명 클래스란?

Java 기반 코드에서 익명 클래스(Anonymous Class)는 말 그대로 클래스 이름이 없기 때문에, 일반적인 클래스 선언 방식과 달리 new 키워드와 함께 바로 클래스 정의를 시작합니다. 즉, 별도의 클래스 파일을 만들거나 이름을 부여하지 않고, 필요한 위치에서 즉석으로 새로운 클래스 형태를 구현하는 것입니다. 이러한 특성은 주로 인터페이스를 구현하거나 추상 클래스를 상속받아 그 안의 추상 메서드를 즉시 오버라이드(Override)해야 할 때 유용하게 쓰입니다.

안드로이드 개발 환경에서는 특히 이벤트 처리와 관련하여 익명 클래스가 자주 활용됩니다. 그 대표적인 예가 View.OnClickListener와 같은 이벤트 리스너 인터페이스 구현입니다. 예를 들어, 버튼을 클릭했을 때 실행할 동작을 정의하려면 보통 setOnClickListener() 메서드를 호출하고, 그 인자로 new View.OnClickListener() { ... } 형태의 익명 클래스를 전달합니다. 이 방식은 이벤트 처리 로직을 해당 UI 요소의 설정 부분에 직접 배치할 수 있어, 코드의 흐름을 이해하기 쉽고 유지보수도 간편합니다. 즉, Java에서 익명 클래스는 불필요한 클래스 선언을 줄이고, 특정 상황에서 한 번만 사용할 구현체를 간결하게 작성할 수 있도록 해 주는 강력한 도구이며, 안드로이드에서는 UI 이벤트 처리 시 그 효율성을 극대화할 수 있는 필수적인 코딩 기법이라 할 수 있습니다. 아래는 익명클래스 기본 구조입니다.

```
new 인터페이스명() {
    // 인터페이스의 추상 메서드 구현
    @Override
    public void 메서드명(...) {
        // 이벤트 발생 시 실행될 코드
    }
};
```

3.1.2 View.OnClickListener 인터페이스와 익명 클래스

안드로이드에서 버튼 클릭과 같은 UI 요소의 클릭 이벤트를 처리하기 위해서는 반드시 View.OnClickListener 인터페이스를 구현해야 합니다. 이 인터페이스는 클릭 동작이 발생했을 때 호

출되는 onClick(View v) 라는 단 하나의 추상 메서드를 포함하고 있으며, 이 메서드 안에 사용자가 클릭했을 때 실행할 구체적인 동작을 작성하게 됩니다. 예를 들어, 버튼을 눌렀을 때 화면 전환을 하거나 메시지를 표시하는 등의 기능은 모두 이 onClick() 메서드 안에서 구현됩니다. 이렇게 이벤트 처리 로직을 인터페이스 구현을 통해 명확하게 분리하면, UI 요소와 그 동작을 구조적으로 관리할 수 있어 코드의 가독성과 유지보수성이 크게 향상됩니다.

아래는 View.OnClickListener 인터페이스 정의를 위한 기본 구조입니다.

```
view.OnclickListener 참조객체변수명 = new View.OnClickListener() {
    void onClick(View v);
}
```

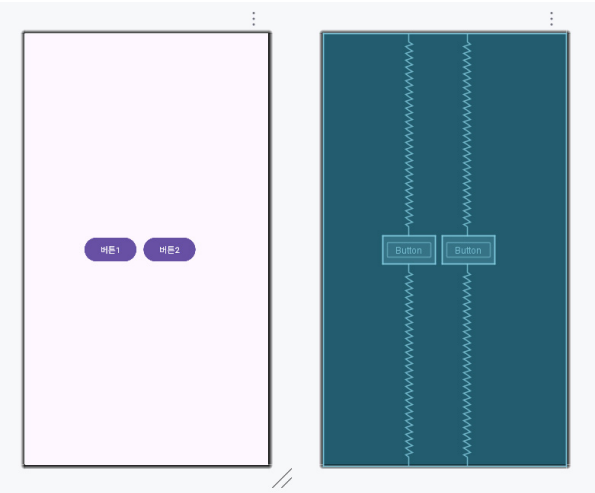
onClick(View v) 메서드는 사용자가 버튼이나 이미지와 같은 UI 요소를 클릭했을 때 자동으로 호출되는 메서드로, 클릭 이벤트를 처리하는 핵심 진입점입니다. 여기서 매개변수(parameter) v는 사용자가 실제로 클릭한 뷰(View) 객체를 참조하며, 이를 통해 어떤 UI 요소에서 이벤트가 발생했는지를 식별할 수 있습니다.

다음은 익명클래스를 이용한 버튼 클릭 이벤트 처리하는 예제입니다.
아래 설정 사항으로 Project를 생성하여 실습해 보세요.

- Project Name : EventExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

layout은 RelativeLayout 으로 변경하고, 속성은 아래와 같이 추가 및 변경 하세요.

뷰	속성값		뷰	속성값	
button	id	btn1	button	id	btn2
	layout_width	wrap_content		layout_width	wrap_content
	layout_height	wrap_content		layout_height	wrap_content
	layout_centerVertical	true		layout_centerVertical	true
	margin_left	100dp		margin_left	200dp
	text	버튼1		text	버튼2



화면 설정 완료 후 "mainActivity.java" 파일에 다음과 같이 코드를 작성하세요.

MainActivity.java

```
package com.example.eventexam;

import android.os.Bundle;
import android.view.View;   // View 클래스
import android.widget.Button;   // Button 클래스
import android.widget.Toast;    // Toast 클래스
import androidx.appcompat.app.AppCompatActivity;

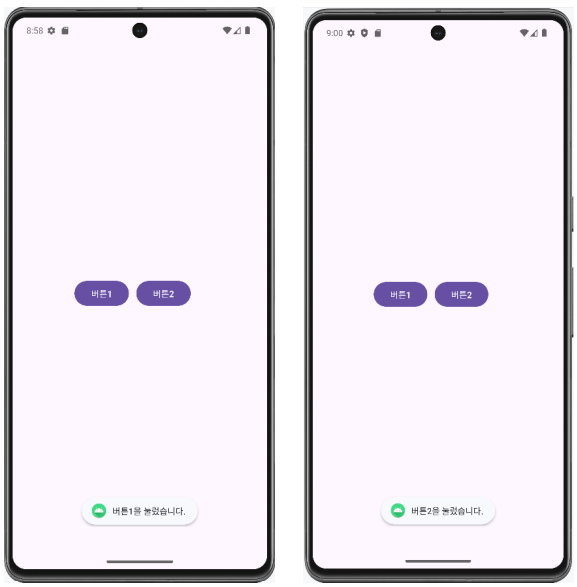
public class MainActivity extends AppCompatActivity {

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

        Button btn1 = (Button) findViewById(R.id.btn1);
        Button btn2 = (Button) findViewById(R.id.btn2);

        btn1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getBaseContext(), "버튼1을 눌렀습니다. ", Toast.LENGTH_LONG).show();
            }
        });
```

```
        btn2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getBaseContext(), "버튼2을 눌렀습니다. ", Toast.LENGTH_
LONG).show();
            }
        });
    }
}
```





코드 설명

- Button btn1 = (Button) findViewById(R.id.btn1);
  화면에 배치된 뷰(View) 중에서 ID가 R.id.btn1로 지정된 버튼을 찾아 Button 타입의 객체 변수 btn1에 할당합니다. 이렇게 함으로써 코드에서 해당 버튼을 제어하거나 이벤트를 연결할 수 있게 됩니다.

- btn1.setOnClickListener(new View.OnClickListener() {

```
        @Override
        public void onClick(View v) {
            ............
        }
    });
```

  btn1 변수에 할당된 버튼이 클릭되었을 때 동작할 이벤트 리스너를 설정합니다. 이때 OnClickListener 인터페이스를 구현하는 익명 클래스를 사용하며, 버튼이 클릭되는 순간 onClick() 메서드가 자동으로 호출되고, 그 안에 작성한 코드가 실행됩니다.

- Toast.makeText(getBaseContext(), "버튼2을 눌렀습니다.",
  Toast.LENGTH_LONG).show();
  화면 하단에 "버튼2을 눌렀습니다." 라는 메시지를 일정 시간 동안 표시합니다.
  Toast는 간단한 안내나 알림을 사용자에게 전달할 때 활용되는 안드로이드의 기본 메시지 출력 방식입니다.

3.1.3 익명 클래스 사용의 장단점

익명 클래스를 활용한 이벤트 처리 방식은 안드로이드 개발에서 자주 사용되는 방법 중 하나로, 분명한 장점과 단점을 모두 가지고 있습니다. 익명 클래스는 별도의 독립된 클래스를 작성할 필요 없이, 이벤트가 필요한 지점에서 즉시 인터페이스를 구현하거나 추상 클래스를 상속 받아 코드를 작성할 수 있기 때문에 코드가 간결해지고 흐름을 따라가기 쉬운 장점이 있습니다. 그러나 반대로, 이벤트 처리 로직이 화면 구성 코드와 밀접하게 섞이게 되면서 코드의 길이가 길어지고 가독성이 떨어질 수 있으며, 재사용성이 떨어진다는 단점도 존재합니다.

그럼에도 불구하고, 화면에 배치된 뷰(View)나 위젯(Widget)의 개수가 많지 않은 비교적 간단한

프로젝트에서는 이러한 단점보다 빠르게 구현할 수 있는 장점이 더 크게 작용하는 경우가 많아, 실무나 학습 단계에서 익명 클래스 방식이 널리 사용됩니다. 아래에 익명 클래스의 장점과 단점을 정리해 두었으니, 앞으로 프로젝트를 진행하며 앱을 개발할 때 이벤트 처리 방법을 선택하는 데 참고하시기 바랍니다. 이를 통해 상황에 맞는 최적의 이벤트 처리 방식을 선택하고, 유지보수성과 효율성을 동시에 고려한 개발을 할 수 있을 것입니다.

- **장점:**
  - 간결성: 이벤트 리스너를 위해 별도의 클래스 파일을 작성할 필요 없이, 필요한 위치에서 바로 정의하고 사용할 수 있으므로 코드가 한층 간결해집니다. 이를 통해 이벤트 처리 로직을 해당 UI 요소와 가까운 위치에 배치할 수 있어, 코드의 흐름을 이해하기도 쉬워집니다.
  - 클로저(Closure) 기능: 익명 클래스 내부에서는 외부 메서드에 선언된 final 또는 사실상 final(effectively final) 변수에 접근할 수 있습니다. 이를 통해 이벤트 처리 로직이 실행될 때, 외부에 존재하는 데이터를 직접 활용할 수 있어 코드 작성이 유연하고 편리해 집니다.

- **단점:**
  - 가독성 저하: 하나의 메서드 안에 여러 개의 익명 클래스가 중첩되어 작성되면, 코드가 길어지고 복잡해져 전체 구조를 한눈에 파악하기 어려워집니다. 이는 유지보수 시 코드 이해를 방해하는 요인이 될 수 있습니다.
  - 재사용성 부족: 익명 클래스는 이름이 없기 때문에 다른 곳에서 재사용하기 어렵습니다. 동일한 로직을 여러 뷰(View)에 적용해야 하는 경우, 각 뷰마다 동일한 익명 클래스 코드를 반복해서 작성해야 하므로 비효율적입니다.
  - 메모리 누수 가능성: 익명 클래스가 외부 액티비티(Activity)에 대한 참조를 암시적으로 유지하는 경우, 익명 클래스의 생명주기가 액티비티의 생명주기보다 길어질 때 메모리 누수가 발생할 수 있습니다. 최근 버전의 안드로이드 스튜디오는 이러한 상황에 대해 경고를 제공하며, 이를 방지하기 위해 약한 참조(WeakReference)를 사용하는 등의 해결 방법을 안내해 주기 때문에 안내에 따라 코드를 수정하면 메모리 누수를 방지할 수 있습니다.

3.2 익명 클래스 참조 객체 생성

익명 클래스는 보통 setOnClickListener(new View.OnClickListener() { ... });와 같이 필요한 위치에서 즉석으로 생성하여 사용됩니다. 이렇게 하면 이벤트 처리 로직을 해당 UI 요소와 가까운 위치에 배치할 수 있어 코드의 흐름을 직관적으로 이해할 수 있다는 장점이 있습니다. 하지만, 동일한 이벤트 처리 로직을 여러 개의 뷰(View)에 적용해야 하는 경우에는 같은 익명 클래스 코드를 각 뷰마다 반복해서 작성해야 하므로 코드가 길어지고 유지보수가 번거로워질 수 있습니다.

이럴 때는 익명 클래스 객체를 변수에 한 번만 할당한 뒤, 이를 여러 뷰에서 재사용하는 방법이 효과적입니다. 예를 들어, View.OnClickListener 타입의 변수를 하나 선언하고, 그 안에 익명 클래스 구현을 담아두면, 버튼, 이미지, 텍스트 뷰 등 서로 다른 UI 요소에 동일한 이벤트 처리를 손쉽게 적용할 수 있습니다. 이렇게 하면 코드 중복을 최소화할 수 있을 뿐만 아니라, 이벤트 처리 로직을 수정해야 할 때도 한 곳만 변경하면 되므로 유지보수성이 크게 향상됩니다.

결국, 익명 클래스는 단순히 한 번만 사용하는 임시 객체로도 유용하지만, 상황에 따라 변수를 활용한 재사용 방식까지 고려한다면 개발 효율성과 코드 품질을 모두 높일 수 있는 유연한 이벤트 처리 방법이 됩니다.

3.2.1 참조 객체 생성 방법

View.OnClickListener 인터페이스를 구현하는 익명 클래스 객체를 생성하여 변수에 할당한 후, 이 변수를 여러 뷰의 리스너로 등록할 수 있습니다.
익명클래스 참조 객체 실습을 위한 Project를 아래와 같이 생성하세요.

- Project Name : EventExam2
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

앞선 세션에서 작업한 "EventExam" 예제와 layout은 동일하게 구성하고, 아래와 같이 MainActivity.java 의 소스 코드만 변경하여 실행해 보세요.

MainActivity.java

```
package com.example.eventexam2;

import android.os.Bundle;
import android.view.View;   // View 클래스
import android.widget.Button;   // Button 클래스
import android.widget.Toast;    // Toast 클래스
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        Button btn1 = (Button) findViewById(R.id.btn1);
        Button btn2 = (Button) findViewById(R.id.btn2);

       // View.OnClickListener 익명 클래스 객체 생성
        View.OnClickListener myClickListener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 클릭된 뷰의 ID를 사용하여 어떤 버튼이 눌렸는지 확인
                String message = "";
                if (v.getId() == R.id.btn1) {
                    message = btn1.getText() + "을 눌렀습니다.";
                } else if (v.getId() == R.id.btn2) {
                    message = btn2.getText() + "을 눌렀습니다.";
                }

                Toast.makeText(getBaseContext(), message, Toast.LENGTH_LONG).show();
            }
        };

       // 생성된 리스너 객체를 여러 버튼에 등록
        btn1.setOnClickListener(myClickListener);
        btn2.setOnClickListener(myClickListener);
    }
}
```

실행 결과는 “EventExam” 실행 결과와 동일하므로, 해당 코드로 변경하여 실행 결과를 비교해 보세요.

코드 설명

- View.OnClickListener myClickListener = new View.OnClickListener() { ... };:
  myClickListener라는 이름의 View.OnClickListener 타입 변수를 선언하고, 여기에 익명 클래스 객체를 할당합니다. 이 객체 안에는 클릭 이벤트가 발생했을 때 실행할 로직이 구현되어 있습니다.
- if (v.getId() == R.id.button1) { ... }:
  onClick(View v) 메서드 내부에서 **v.getId()**를 호출하여 사용자가 클릭한 뷰의 ID 값을 가져옵니다. 이 값을 미리 정의된 버튼 ID(R.id.button1, R.id.button2 등)와 비교하면, 어떤 버튼이 클릭되었는지 식별할 수 있습니다. 이를 통해 버튼별로 다른 동작을 구현할 수 있습니다.
- btn1.setOnClickListener(myClickListener);:
  btn1에 동일한 myClickListener 객체를 클릭 이벤트 리스너로 등록합니다. 이렇게 하면 여러 개의 버튼이 같은 이벤트 처리 로직을 공유할 수 있어, 코드 중복을 줄이고 유지보수성을 높일 수 있습니다.

3.2.2 익명클래스 참조 객체 장점 및 활용

- **장점:**
  - 동일한 이벤트 처리 로직을 하나의 익명 클래스 객체로 구현해 두면, 이를 여러 뷰(View)에 손쉽게 적용할 수 있습니다. 이렇게 하면 각 뷰마다 동일한 코드를 반복해서 작성할 필요가 없어 중복을 줄이고 재사용성을 높일 수 있습니다.
  - 코드 관리 용이: 이벤트 처리 로직이 한 곳에 집중되어 있기 때문에, 기능을 수정하거나 개선해야 할 때 해당 부분만 변경하면 됩니다. 이로 인해 유지보수가 훨씬 간편해집니다.
  - 가독성 향상: 각 뷰에 setOnClickListener()를 호출할 때 긴 코드 블록을 직접 작성하는 대신, 미리 정의된 리스너 객체를 등록하는 방식이므로 코드가 간결해지고 읽기 쉬워집니다.

- **활용:**
  - 해당 방식은 여러 개의 버튼이 비슷한 동작을 수행하지만, 버튼에 따라 세부 동작만 약

간씩 다른 경우에 특히 유용합니다. 예를 들어, 숫자 키패드와 같이 0부터 9까지의 숫자 버튼이 모두 동일한 OnClickListener를 사용하도록 설정하고, onClick(View v) 메서드 안에서 v.getId()를 이용해 어떤 버튼이 눌렸는지를 구분할 수 있습니다. 이렇게 하면 각 버튼이 개별적으로 이벤트 코드를 갖는 것이 아니라, 하나의 공통 로직을 공유하면서도 버튼별로 필요한 동작 차이를 구현할 수 있습니다.

결국, 이 방법은 코드 구조를 단순화하고 유지보수성을 높이며, 특히 동일 패턴의 UI 요소가 다수 존재하는 프로젝트에서 큰 장점을 발휘합니다.

3.3 인터페이스 상속 (Activity에 구현)

익명 클래스를 활용한 이벤트 처리 방식 외에도, 액티비티(Activity) 클래스 자체가 이벤트 리스너 인터페이스를 직접 구현(상속)하는 방법이 있습니다. 이 방식에서는 액티비티가 해당 인터페이스의 추상 메서드를 구현함으로써, 화면에 배치된 여러 개의 뷰(View)에서 발생하는 이벤트를 하나의 중앙화된 메서드에서 일괄적으로 처리할 수 있습니다. 이렇게 하면 각 뷰마다 별도의 익명 클래스나 리스너 객체를 생성할 필요가 없어, 코드의 구조가 단순해지고 전체적인 가독성이 향상됩니다. 특히, 액티비티가 다수의 버튼이나 위젯의 이벤트를 관리해야 하는 경우, 이 방법을 사용하면 이벤트 처리 로직이 한 곳에 모이게 되어 유지보수와 수정 작업이 훨씬 용이해집니다.

또한, 이 방식은 이벤트 처리 흐름을 이해하기에도 직관적입니다.

예를 들어, "MainActivity"가 View.OnClickListener 인터페이스를 구현하도록 하고, onClick(View v) 메서드 안에서 v.getId()를 통해 클릭된 뷰를 식별한 뒤, 그에 맞는 동작을 수행하도록 작성하면 됩니다. 이처럼 액티비티 자체를 이벤트 리스너로 설정하는 방식은 특히 UI 요소의 개수가 많고 이벤트 로직이 서로 관련이 있는 경우에 유용하게 활용될 수 있습니다.

3.3.1 Activity에 View.OnClickListener 구현하기

액티비티 클래스 선언부에 implements View.OnClickListener를 추가하고, onClick(View v) 메서드를 오버라이드합니다.
아래와 같이 Project를 생성하세요.

- Project Name : EventExam3

- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

레이아웃은 "EventExam", "EventExam2" Project 와 동일하니, 이전 Project를 참조하여 레이아웃을 구성하고, 아래 코드를 작성하세요.

MainActivity.java

```
package com.example.eventexam3;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

// MainActivity가 View.OnClickListener 인터페이스를 구현하도록 선언
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    // Button 객체를 전역변수로 선언
    private TextView statusTextView;
    private Button btn1, btn2;

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

        statusTextView = findViewById(R.id.statusTextView);
        btn1 = findViewById(R.id.btn1);
        btn2 = findViewById(R.id.btn2);

        // 각 버튼에 현재 액티비티(this)를 클릭 리스너로 등록
        btn1.setOnClickListener(this);
        btn2.setOnClickListener(this);
    }

    // View.OnClickListener 인터페이스의 onClick 메서드를 오버라이드
    @Override
    public void onClick(View v) {
        // 클릭된 뷰의 ID를 사용하여 어떤 버튼이 눌렸는지 확인
        String message = "";
        int viewId = v.getId(); // 클릭된 뷰의 ID를 가져옴
```

```
            if (viewId == R.id.btn1) {
                message = btn1.getText() + "을 눌렀습니다.";
            } else if (viewId == R.id.btn2) {
                message = btn2.getText() + "을 눌렀습니다.";
            } else {
                message = "버튼이 아닙니다."; // 혹시 모를 경우 대비
            }

            Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
        }
    }
```

실행 결과는 "EventExam" 실행 결과와 동일하므로, 해당 코드로 변경하여 실행 결과를 비교해 보세요.



코드 설명

- public class MainActivity extends AppCompatActivity implements View.OnClickListener:
  MainActivity 클래스가 AppCompatActivity를 상속받고, 동시에 View.OnClickListener 인터페이스를 구현하도록 선언합니다. 이를 통해 액티비티 자체가 클릭 이벤트 리스너 역할을 수행할 수 있습니다.
- btn1.setOnClickListener(this);:
  setOnClickListener() 메서드의 인자로 this를 전달하여 현재 액티비티를 이벤트 리스너로 등록합니다. 이는 MainActivity가 View.OnClickListener를 구현하고 있기 때문에 가능한 방식입니다.
- @Override public void onClick(View v):
  View.OnClickListener 인터페이스의 추상 메서드인 onClick(View v)를 MainActivity 클래스 내에서 직접 구현합니다. 이렇게 하면 화면의 여러 뷰에서 발생하는 모든 클릭 이벤트가 이 하나의 메서드로 모이게 되어, 이벤트 처리를 일원화할 수 있습니다.
- int viewId = v.getId();:
  매개변수 v를 통해 클릭된 뷰 객체를 참조하고, getId() 메서드를 사용하여 해당 뷰의 ID 값을 가져옵니다. 이렇게 얻은 ID를 기준으로 if-else if 문이나 switch 문을 활용하여 어떤 뷰가 클릭되었는지를 구분하고, 각 뷰에 맞는 동작 로직을 실행합니다.

3.3.2 장단점 및 활용

• **장점:**

– 코드의 간결성: 여러 개의 뷰(View)에 대한 이벤트 처리 로직을 하나의 onClick 메서드 안에 통합할 수 있어, 코드 구조가 훨씬 깔끔해집니다. 이로 인해 각 뷰마다 별도의 익명 클래스를 생성할 필요가 없어 중복이 줄어듭니다.

– 메모리 효율성: 익명 클래스 방식처럼 이벤트 발생 시마다 새로운 객체를 생성하지 않고, 액티비티(Activity) 객체 하나가 모든 클릭 이벤트를 처리하므로 메모리 사용 면에서 보다 효율적일 수 있습니다.

– 액티비티 멤버 접근 용이: onClick 메서드가 액티비티 클래스 내부에 위치하므로, 액티비티의 멤버 변수나 메서드에 직접 접근하기가 쉽습니다. 이를 통해 이벤트 처리 로직에서 필요한 데이터나 기능을 빠르게 호출할 수 있습니다.

• **단점:**

– onClick 메서드의 복잡성 증가: 처리해야 할 뷰의 종류가 많아질수록 onClick 메서드 내부의 if-else if 문이나 switch 문이 길어지고 복잡해져 코드 가독성이 떨어질 수 있습니다.

– 모든 클릭 이벤트의 집중화: 액티비티 내의 모든 클릭 이벤트가 하나의 메서드에 집중되므로, 이벤트가 많아질 경우 로직을 기능별로 분리하거나 관리하기가 어려워질 수 있습니다.

• **활용:**

– 액티비티에서 비교적 적은 수의 뷰에 대한 클릭 이벤트를 처리할 때 적합합니다.

– 여러 뷰가 동일한 onClick 로직을 공유하되, v.getId()를 사용하여 어떤 뷰에서 이벤트가 발생했는지 식별해야 할 때 유용하게 사용할 수 있습니다.

3.4 위젯 onClick 속성 활용

안드로이드에서는 XML 레이아웃 파일에서 각 위젯의 속성 중 하나인 android :onClick 속성을 활용하여, 해당 위젯이 클릭되었을 때 실행될 메서드를 직접 지정할 수 있습니다. 이 방법을 사용하면 별도의 setOnClickListener() 호출 없이도, XML에서 지정한 메서드 이름과 동일한 이름의 메서드를 액티비티(Activity) 클래스에 구현하기만 하면 클릭 이벤트 처리가 이루어집니다.

이 방식의 가장 큰 장점은 구현이 매우 간결하다는 점입니다. 이벤트 리스너 객체를 생성하거나 등록하는 코드가 필요 없기 때문에, XML과 자바(Java) 또는 코틀린(Kotlin) 코드 간의 매핑이 직관적이고 빠르게 이루어집니다. 특히, 간단한 버튼 클릭과 같은 단순 이벤트를 처리할 때는 코드 양을 크게 줄일 수 있어 개발 속도를 높이는 데 유리합니다.

하지만 이 방법에는 몇 가지 제약 조건이 존재합니다. 우선, android:onClick 속성으로 지정하는 메서드는 반드시 public 접근 제한자를 가져야 하고, 반환형이 void이며, View 타입의 매개변수 하나를 가져야 합니다. 또한 해당 메서드는 반드시 XML 레이아웃을 로드하는 액티비티나 프래그먼트 클래스 내부에 존재해야 하며, 다른 클래스에서 구현된 메서드를 직접 연결할 수 없습니다. 이러한 제약으로 인해 복잡한 이벤트 처리나 여러 위젯 간의 동적 제어가 필요한 경우에는 사용이 적합하지 않을 수 있습니다.

결론적으로, android:onClick 속성 방식은 단순하고 빠른 이벤트 연결이 필요한 경우에 적합하지만, 프로젝트 규모가 커지고 이벤트 처리 로직이 복잡해질수록 다른 이벤트 처리 방식과 병행하거나 대체하는 것이 바람직합니다.

3.4.1 android:onClick 속성 사용 방법

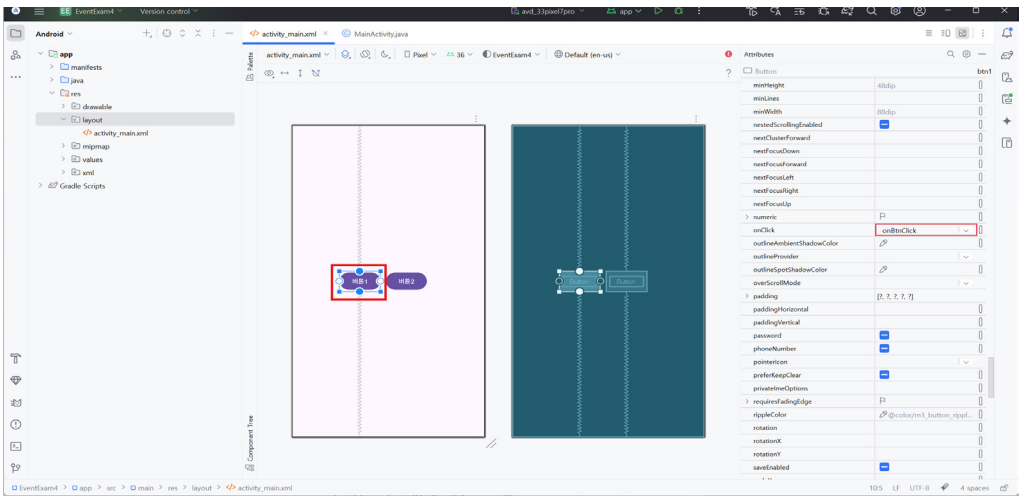
안드로이드의 버튼이나 다른 클릭 가능한 뷰(View)에는 android:onClick라는 속성이 있으며, XML 레이아웃에서 이 속성에 실행할 메서드 이름을 지정할 수 있습니다. 이후, 해당 메서드 이름과 동일한 이름의 메서드를 액티비티(Activity) 내에 정의하면, 뷰가 클릭될 때 자동으로 그 메서드가 호출되어 이벤트를 처리할 수 있습니다.
아래와 같이 Project를 생성하세요.

- Project Name : EventExam4
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

레이아웃은 "EventExam", "EventExam2", "EventExam3" Project 와 동일하니, 이전 Project를 참조하여 레이아웃을 구성하고, 순서에 따라 프로젝트를 작성하세요.

아래와 같이 레이아웃 설정화면에서 Button 뷰/위젯의 android:onClick 속성 값을 onBtnClick으로 입력하세요.

뷰	속성값		뷰	속성값	
button	id	btn1	button	id	btn2
	layout_width	wrap_content		layout_width	wrap_content
	layout_height	wrap_content		layout_height	wrap_content
	layout_centerVertical	true		layout_centerVertical	true
	margin_left	100dp		margin_left	200dp
	text	버튼1		text	버튼2
	android:onClick	onBtnClick		android:onClick	onBtnClick



위 그림은 btn1을 설정하는 화면입니다. 이 화면을 참고하여 동일한 방식으로 btn2를 설정해 보세요.

MainActivity.java

```
package com.example.eventexam3;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
```

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

    /**
     * XML의 android:onClick 속성에 지정된 메서드.
     * 이 메서드는 반드시 public void 이고, View 타입의 매개변수 하나를 가져야 한다.
     * @param view 클릭된 뷰 객체
     */
    public void onBtnClick(View view) {
        // 이 메서드는 버튼이 클릭되었을 때 호출된다.
        String message = "";
        int viewId = view.getId(); // 클릭된 뷰의 ID를 가져옴

        if (viewId == R.id.btn1) {
            message = "버튼1을 눌렀습니다.";
        } else if (viewId == R.id.btn2) {
            message = "버튼2를 눌렀습니다.";
        } else {
            message = "버튼이 아닙니다."; // 혹시 모를 경우 대비
        }

        Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
    }
}
```

실행 결과는 "EventExam" 실행 결과와 동일하므로, 해당 코드로 실행 결과를 비교해 보세요.



코드 설명

- public void onXmlButtonClick(View view):
  MainActivity 클래스 내에 **onXmlButtonClick**이라는 이름의 메서드를 정의합니다. 이 메서드는 android:onClick 속성을 통해 XML 레이아웃과 연결되며, 다음과 같은 조건을 반드시 충족해야 합니다.
  - public 접근 제어자: 메서드는 외부에서 접근 가능해야 하므로 public으로 선언해야 합니다.
  - void 반환 타입: 메서드는 어떤 값도 반환하지 않아야 합니다.
  - View 타입의 매개변수 하나: 클릭된 뷰 객체를 전달받기 위해 View 타입의 매개변수 하나를 선언해야 합니다. 매개변수의 이름은 view가 아니어도 상관없지만, 관례적으로 view 또는 v를 많이 사용합니다.

3.4.2 위젯 onClick 장점 및 단점

• **장점:**

– 매우 간결한 코드: XML에서 직접 이벤트 처리 메서드를 지정할 수 있어, 자바(Java)나 코틀린(Kotlin) 코드에서 findViewById()를 호출하고 setOnClickListener()를 설정하는 과정을 생략할 수 있습니다.

– 빠른 프로토타이핑: 간단한 클릭 이벤트를 처리할 때 매우 빠르게 적용할 수 있어, 초기 화면 설계나 프로토타입 제작 시 유용합니다.

• **단점:**

– 제약 조건 존재: 메서드의 접근 제어자, 반환 타입, 매개변수 타입이 엄격하게 제한되며, 이 조건을 충족하지 않으면 앱 실행 시 오류가 발생합니다.

– 하나의 뷰에 하나의 리스너만 가능: android:onClick 속성은 한 뷰에 하나의 클릭 리스너만 연결할 수 있어, 여러 이벤트 리스너를 동시에 등록해야 하는 복잡한 상황에는 적합하지 않습니다.

– 코드와 레이아웃 간의 결합도 증가: 이벤트 처리 메서드의 이름이 XML에 직접 노출되기 때문에 코드와 레이아웃의 분리도가 낮아지고, 이로 인해 리팩토링 시 변경 작업이 번거로워질 수 있습니다.

– 테스트의 어려움: 단위 테스트 환경에서 android:onClick으로 연결된 메서드를 직접 호출하거나 검증하기가 어렵습니다.

– 적용 범위 제한: android:onClick 속성은 OnClickListener와 같은 클릭 이벤트에만 적용할 수 있으며, OnLongClickListener(길게 누르기), OnTouchListener(터치), OnKeyListener(키 입력) 등 다른 이벤트 유형에는 사용할 수 없습니다.

3.4.3 언제 android:onClick을 사용할까?

android:onClick 속성은 다음과 같은 경우에 고려해 볼 수 있습니다.

- 매우 간단하고 독립적인 클릭 이벤트 로직을 가진 뷰(예: "닫기" 버튼, "확인" 버튼 등)
- 프로토타입이나 간단한 앱을 빠르게 만들 때
- 하나의 뷰에 하나의 클릭 이벤트만 필요할 때

하지만 앱의 규모가 크거나 기능이 복잡해져서 여러 개의 뷰(View)가 동일한 로직을 공유해야 하거나, 클릭 이벤트뿐만 아니라 다양한 종류의 이벤트(예: 길게 누르기, 터치, 키 입력 등)를 함께 처리해야 하는 상황이라면, android:onClick 속성만으로는 한계가 존재합니다. 이러한 경우에는 익명 클래스를 사용하여 각 뷰에 맞는 이벤트를 개별적으로 처리하거나, 액티비티(Activity)나 별도의 클래스가 이벤트 리스너 인터페이스를 직접 구현하는 방식을 활용하는 것이 더 유연합니다.

이러한 방법은 이벤트 처리 로직을 상황별로 명확하게 분리할 수 있고, 코드와 레이아웃 간의 결합도를 낮춰 유지보수와 확장성 측면에서도 훨씬 유리합니다. 특히, 복잡한 앱에서는 동일한 로직을 여러 뷰에서 재사용하거나, 이벤트 유형에 따라 다른 처리 흐름을 적용해야 하는 경우가 빈번하므로, 인터페이스 구현 방식이나 익명 클래스를 적절히 활용하면 코드 구조를 체계적으로 유지할 수 있고, 기능 변경이나 추가가 필요한 경우에도 최소한의 수정만으로 대응할 수 있습니다.

3.5 이벤트 처리 방법 선택 가이드

지금까지 안드로이드에서 이벤트를 처리하는 세 가지 주요 방법에 대해 학습했습니다. 각 방법은 고유한 장단점을 가지고 있으므로, 개발 상황과 요구 사항에 따라 적절한 방법을 선택하는 것이 중요합니다.

표 3-1 상황에 맞는 이벤트 선정

이벤트 방식	내 용	
익명 클래스	• setOnClickListener(new View.OnClickListener() { ... }); 형태로 즉석에서 리스너 객체 생성	
	장점	• 간결하게 이벤트 처리 로직 정의 가능 • 외부변수 접근 용이
	단점	• 재사용성 부족
	활용 상황	• 코드 중복 발생 가능 • 가독성 저하 (중첩시)
	비고	• 단일 뷰의 독립적인 이벤트 처리 • 특정 뷰에만 적용되는 복잡한 로직
익명 클래스 참조 객체	• View.OnClickListener listener = new View.OnClickListener() { ... }; 형태로 객체 생성 후 여러 뷰에 등록	
	장점	• 코드 재 사용성 높음 • 코드 관리 용이 • 가독성 향상
	단점	• onClick 메서드 내에서 뷰 식별 로직 필요 (if~else if 또는 switch)
	활용 상황	• 여러 뷰가 동일하거나 유사한 이벤트 처리 로직을 가질때

인터페이스 상속 (Activity 구현)	• Activity implements View.OnClickListener 후 this를 리스너로 등록	
	장점	• 코드의 간결성(액티비티 내 통합) • 메모리 효율성 • 액티비티 멤버 접근 용이
	단점	• onClick 메서드 복잡성 증가 (뷰 많아질수록) • 모든 클릭 이벤트가 한 곳으로 집중
	활용 상황	• 액티비티 내에서 비교적 적은 수의 뷰 이벤트 처리 • 뷰 식별을 통해 로직 분기
andorid:onClick 속성	• XML 레이아웃에서 android:onClick="메서드명"으로 지정	
	장점	• 매우 간결한 XML 코드 • 빠른 프로토타이핑
	단점	• 엄격한 메서드 제약조건 • 하나의 뷰에 하나의 클릭 리스너만 가능 • 다른 종류의 이벤트에는 사용불가 • 코드와 레이아웃 결합도 증가
	활용 상황	• 매우 간단하고 독립적인 클릭 이벤트 • 빠른 구현이 중요할때

표에서 정리한 각 방법의 장단점과 활용 상황을 참고하여, 개발자가 스스로 상황에 맞는 방식을 선택할 수 있지만, 일반적으로 다음과 같은 권장 사항을 따르는 것이 좋습니다.

- 단일 뷰의 간단한 클릭 이벤트: 익명 클래스를 사용하거나, 더 간결하게 작성하고자 할 경우 android:onClick 속성을 고려합니다. 특히 코틀린(Kotlin)에서는 람다식을 사용하면 코드가 더욱 짧아집니다.
- 여러 뷰가 유사한 클릭 이벤트를 공유하는 경우: 익명 클래스 객체를 변수로 선언해 재사용하거나, 액티비티에 View.OnClickListener를 구현한 뒤 v.getId() 또는 v.getTag()를 통해 클릭된 뷰를 식별하는 방식을 사용하는 것이 효율적입니다.
- 복잡한 이벤트 처리 또는 다양한 이벤트 유형이 필요한 경우: 익명 클래스를 상황에 맞게 사용하거나, 별도의 클래스를 정의해 리스너를 구현하는 방식이 가장 유연하고 확장성이 높습니다.

또한, 현대의 안드로이드 개발에서는 코틀린(Kotlin) 언어의 람다식(Lambda Expression)을 활용하여 익명 클래스 구현을 훨씬 더 간결하게 작성할 수 있습니다. 예를 들어, setOnClickListener를 사용할 때 기존 자바 방식과 달리 코틀린에서는 다음과 같이 단 한 줄로 표현할 수 있습니다.

MainActivity.java

```
// Kotlin 람다식을 이용한 버튼 클릭 이벤트 처리
clickButton.setOnClickListener {
    // 버튼이 클릭되었을 때 실행될 코드
    Toast.makeText(this, "클릭 이벤트 발생!", Toast.LENGTH_SHORT).show()
}
```

이는 자바에서 사용하던 익명 클래스 방식을 람다식으로 변환한 형태로, 코드의 가독성을 높이고 작성 과정을 훨씬 간결하게 만들어 줍니다. 자바 8 이상에서도 람다식을 사용할 수 있지만, 코틀린(Kotlin)에서는 기본 문법으로 람다식을 적극적으로 활용하여 이벤트 처리 코드를 더욱 직관적으로 표현합니다.

따라서, 이 챕터에서 학습한 다양한 이벤트 처리 방식을 충분히 이해하고, 실제 앱 개발 과정에서 직접 적용해 보면서 각 방법이 가지는 장점과 단점을 체득하는 것이 중요합니다. 이를 통해 상황에 맞는 최적의 이벤트 처리 방식을 선택하고, 보다 효율적이고 유지보수하기 쉬운 코드를 작성할 수 있게 될 것입니다.



연습문제

1. 안드로이드에서 이벤트(Event)란 무엇이며, 이벤트 처리가 왜 중요한지 설명하시오.

2. 안드로이드에서 이벤트를 처리하는 방법 중 익명 클래스 방식과 참조 객체 방식을 비교하여 장단점을 설명하시오.

3. 개발자가 모든 사용자 동작(예: 클릭, 더블클릭, 롱클릭 등)에 대해 이벤트 처리 코드를 작성하지 않고 특정 이벤트만 처리하는 이유를 사용자 경험 관점에서 논하시오.

4. 안드로이드에서 이벤트 리스너(Event Listener)의 역할을 설명하고, 버튼 클릭 이벤트를 처리하기 위한 기본적인 리스너 설정 방법을 서술하시오.

5. 화면에 버튼 하나가 있다고 가정할 때, 사용자가 버튼을 눌렀을 때 특정 메시지를 출력하는 기능을 구현하려면 어떤 이벤트 리스너를 사용해야 하며, 그 이유를 설명하시오.



제4장

메뉴와 대화상자



학습목표

- 옵션 메뉴, 컨텍스트 메뉴, 팝업 메뉴 차이와 설명할 수 있다.
- AlertDialog로 확인, 선택, 입력형 대화상자를 구현할 수 있다.
- DatePickerDialog, TimePickerDialog 등 적절한 대화상자 패턴을 적용할 수 있다.
- 커스텀 레이아웃 대화상자를 구성하고 제어할 수 있다.

안드로이드 앱에서 메뉴(Menu)와 대화상자(Dialog)는 사용자가 앱을 보다 효율적으로 활용할 수 있도록 추가적인 기능을 제공하거나, 중요한 정보를 전달하며, 특정 동작에 대한 결정을 유도하는 데 사용되는 핵심 UI 요소입니다. 메뉴는 화면 상단의 옵션 메뉴나 하단의 컨텍스트 메뉴 등 다양한 형태로 제공되어, 사용자가 현재 화면에서 접근 가능한 기능을 빠르게 선택할 수 있도록 돕습니다. 반면, 대화상자는 화면의 일부를 차지하는 팝업 형태로 나타나며, 간단한 알림 메시지부터 사용자 입력, 확인 및 취소와 같은 선택 동작까지 폭넓게 활용됩니다.

이러한 메뉴와 대화상자는 단순히 시각적인 요소를 넘어서, 앱의 흐름을 제어하고 사용자 경험(UX)을 향상시키는 중요한 역할을 합니다. 예를 들어, 메뉴를 통해 숨겨진 기능에 접근하거나, 대화상자를 통해 중요한 경고 메시지를 표시하고 사용자의 의사를 묻는 등, 앱의 상호작용성을 높일 수 있습니다.

따라서 이 챕터에서는 안드로이드 앱에서 제공되는 다양한 종류의 메뉴와 대화상자를 구현하고 사용하는 방법을 체계적으로 학습합니다. 각 UI 요소의 특성과 용도, 구현 절차뿐만 아니라 상황에 맞는 활용 방법까지 익혀, 실제 앱 개발에서 사용자 친화적이고 기능적인 UI를 설계할 수 있는 역량을 기르는 것을 목표로 합니다.

4.1 메뉴

안드로이드 앱에서는 주로 세 가지 종류의 메뉴가 활용되며, 각각의 메뉴는 사용되는 상황과 목적이 서로 다릅니다. 첫 번째는 옵션 메뉴(Options Menu)로, 일반적으로 화면 상단의 앱 바(App Bar)나 툴바(Toolbar)에 위치하며, 앱 전반에서 자주 사용되는 주요 기능이나 설정 항목을 제공합니다. 두 번째는 컨텍스트 메뉴(Context Menu)로, 특정 뷰(View)를 길게 누르는 등의 특정 동작을 통해 표시되며, 해당 뷰나 항목과 직접적으로 연관된 기능을 제공합니다. 마지막으로 팝업 메뉴(Popup Menu)는 화면의 특정 위치에 작게 나타나는 메뉴로, 사용자의 현재 작업 맥락에 맞춰 빠르게 실행할 수 있는 기능을 제시합니다.

이처럼 각 메뉴 유형은 노출 시점, 표시 위치, 제공 기능의 범위에서 차이를 가지며, 적절한 상황에 맞춰 사용하면 앱의 기능 접근성을 높이고 사용자 경험을 향상시킬 수 있습니다.

4.1.1 옵션 메뉴(Option Menu)

옵션 메뉴(Options Menu)는 안드로이드 앱에서 가장 일반적으로 사용되는 메뉴 형식으로, 주로 화면 상단의 타이틀 바(Title Bar)나 앱 바(App Bar)에 위치한 메뉴 버튼을 클릭하거나, 특정 기기에서는 하드웨어 메뉴 버튼을 눌렀을 때 나타납니다. 이 메뉴는 앱 전반에서 자주 사용되는 주요 기능, 설정 항목, 전역 명령 등을 제공하는 용도로 사용되며, 사용자가 현재 보고 있는 화면과 관계없이 앱의 공통 기능에 쉽게 접근할 수 있도록 설계됩니다.

예를 들어, 옵션 메뉴에는 환경 설정(Settings), 도움말(Help), 검색(Search)과 같이 앱 어디서나 호출 가능한 기능들이 포함될 수 있습니다. 사용자는 메뉴 버튼을 클릭하면 드롭다운 형태로 항목들이 표시되며, 원하는 기능을 손쉽게 선택할 수 있습니다. 이러한 구조는 앱의 주요 기능을 직관적으로 제공하여, 기능 접근성을 높이고 사용자 경험을 향상시키는 데 중요한 역할을 합니다.

옵션 메뉴 실습을 위하여 아래와 같이 설정하여 Project를 생성하세요.

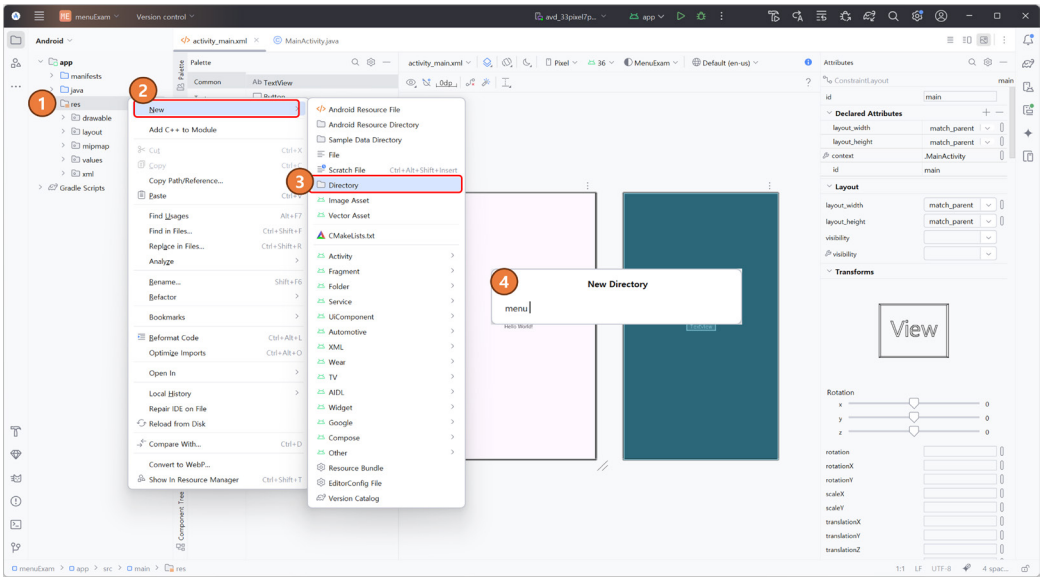
- Project Name : menuExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

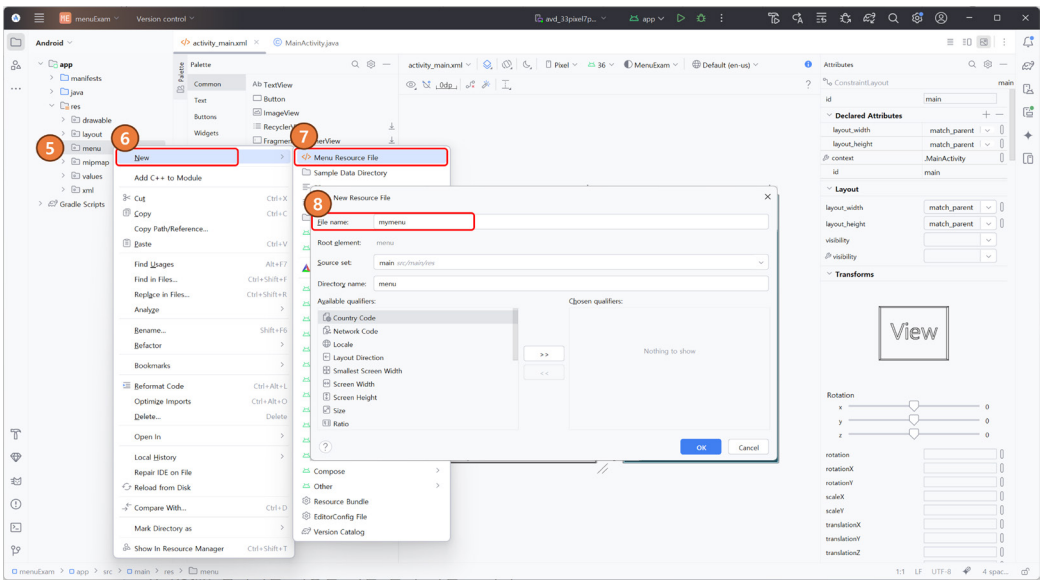
메뉴를 정의하기 위해서는 먼저 XML 리소스 파일을 생성해야 합니다. 이 파일은 res/menu 폴더에 위치합니다.

- res/menu 폴더 생성: Project Navigator(프로젝트 뷰)에서 ❶ app 〉 res 폴더를 우클릭 ❷ New 〉 ❸ Directory를 선택하고, ❹ 폴더명을 menu 로 입력한다.
- 메뉴 XML파일 생성: 생성된 ❺ menu 폴더를 우클릭 〉 ❻ New 〉 ❼ Menu Resource File 을 선택합니다. 파일 이름을 ❽ mymenu 로 지정하고 OK를 클릭합니다.

메뉴 파일의 기본 구조는 아래와 같습니다.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <!-- 메뉴 아이템들이 여기에 추가됩니다. -->
</menu>
```





메뉴 아이템 정의 ⟨item⟩ 태그: 〈item〉 태그를 사용하여 각 메뉴 항목을 정의합니다.

메뉴 파일명.xml

```
<!-- 메뉴 파일 예시 -->
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/action_settings"
        android:icon="@android:drawable/ic_menu_preferences"
        android:title="설정"
        app:showAsAction="never" /> <!-- 항상 오버플로우 메뉴에 표시 -->

    <item
        android:id="@+id/action_search"
        android:icon="@android:drawable/ic_menu_search"
        android:title="검색"
        app:showAsAction="ifRoom" /> <!-- 공간이 있으면 액션바에, 없으면 오버플로우에 -->

    <item
        android:id="@+id/action_share"
        android:icon="@android:drawable/ic_menu_share"
        android:title="공유"
        app:showAsAction="always" /> <!-- 항상 액션바에 표시 -->

    <item
        android:id="@+id/action_exit"
        android:title="종료"
        app:showAsAction="never" />

</menu>
```

메뉴 파일 예시와 같이 메뉴 항목을 구성하려면 반드시 〈item〉 태그를 사용해야 합니다. 이 〈item〉 태그는 각 메뉴 항목을 정의하는 데 사용되며, 항목의 식별자, 표시되는 텍스트, 아이콘, 동작 방식 등 다양한 속성을 지정할 수 있습니다. 아래에서는 〈item〉 태그에서 자주 사용되는 주요 속성과 그 역할을 정리하였습니다.

표 4-1 〈item〉태그 주요속성

〈item〉 태그 주요 속성	설 명	
android:id	메뉴 아이템을 고유하게 식별하는 ID입니다. 코드에서 이 ID를 사용하여 메뉴 클릭을 처리합니다.	
android:icon	메뉴 아이템 옆에 표시될 아이콘을 지정합니다. (drawable 리소스 참조)	
android:title	메뉴 아이템에 표시될 텍스트입니다. (문자열 리소스 참조 권장: @string/app_name 등)	
app:showAsAction	액션바(ActionBar) 또는 툴바(Toolbar)에 어떻게 표시될지 결정합니다.	
	never	항상 오버플로우 메뉴 (점 3개 아이콘) 안에 표시됩니다.
	ifRoom	액션바에 공간이 있을 경우 표시되고, 없으면 오버플로우 메뉴로 이동합니다.
	always	항상 액션바에 표시됩니다. (공간이 부족해도 강제로 표시될 수 있으므로 주의)
	withText	ifRoom 또는 always와 함께 사용될 때, 아이콘과 함께 텍스트도 표시됩니다.
	collapseActionView	액션 뷰(Action View)가 접힐 수 있도록 합니다. (예: 검색 위젯)
android:checkable	메뉴 아이템이 체크 가능한지 여부를 지정합니다. (true 또는 false).	
android:checked	메뉴 아이템이 기본적으로 체크되어 있는지 여부를 지정합니다. (true 또는 false).	
android:menuCategory	메뉴 아이템의 그룹 카테고리를 지정합니다. (예: container, system, secondary, alternative)	
android:orderInCategory	메뉴 그룹 내에서 아이템의 정렬 순서를 지정합니다. 숫자가 작을수록 먼저 표시됩니다.	
android:alphabeticShortcut	메뉴 아이템에 대한 영문 단축키를 지정합니다. (예: android:alphabeticShortcut="s")	
android:numericShortcut	메뉴 아이템에 대한 영문 단축키를 지정합니다. (예: android:alphabeticShortcut="s")	
android:actionLayout	메뉴 아이템에 대한 사용자 정의 뷰 클래스를 지정합니다. (예: SearchView)	
android:actionViewClass	메뉴 아이템에 대한 사용자 정의 뷰 클래스를 지정합니다. (예: SearchView)	

안드로이드 앱에서 메뉴를 구성할 때, 기능적으로나 의미적으로 서로 관련이 있는 메뉴 아이템들을 하나의 그룹으로 묶어 두면 코드의 가독성을 높이고 유지보수를 훨씬 더 효율적으로 할 수 있습니다. 이러한 경우에 사용하는 태그가 바로 〈group〉 태그입니다.

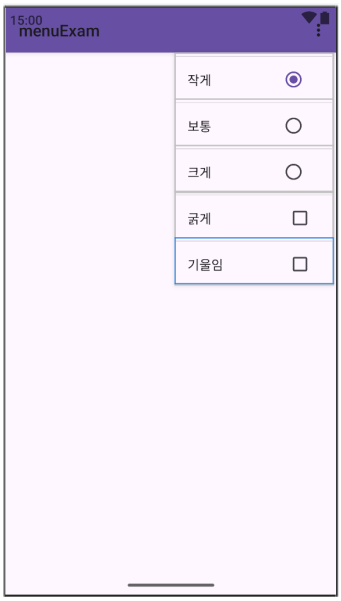


그림 4-1 〈group〉 태그 활용한 메뉴구성 예시

〈group〉 태그는 여러 개의 〈item〉 메뉴 항목을 논리적으로 하나의 묶음으로 구성할 수 있도록 해 주며, 이렇게 묶인 그룹은 공통 속성을 공유할 수 있습니다. 예를 들어, 그룹 전체에 동일한 가시성(visible) 상태를 적용하거나, 한 번에 체크 가능 여부(checkable)를 설정하는 등의 관리가 가능합니다. 또한 〈group〉 태그를 사용하면 메뉴 아이템의 동작 방식이나 표시 속성을 일괄적으로 변경할 수 있어, 관련 기능들이 많은 대규모 앱에서 특히 유용합니다.

예를 들어, 설정(Settings) 화면에서 '보기(View) 옵션'에 해당하는 여러 메뉴 항목을 하나의 그룹으로 묶어두면, 사용자가 한 메뉴를 선택했을 때 다른 메뉴의 체크 상태를 자동으로 해제하거나, 상황에 따라 그룹 전체를 숨기는 등의 처리를 손쉽게 구현할 수 있습니다.

아래는 〈group〉 태그를 사용하여 메뉴를 구성한 XML 예시로, 이 구조를 이해하면 메뉴 설계 시 보다 체계적이고 유지보수가 용이한 코드 작성이 가능합니다.

메뉴 파일명.xml

```
<!-- <group> 태그 메뉴 예시 -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:id="@+id/font_size_group" android:checkableBehavior="single">
        <item
            android:id="@+id/font_small"
            android:title="작게"
            android:checked="true" />
        <item
            android:id="@+id/font_medium"
            android:title="보통" />
        <item
            android:id="@+id/font_large"
            android:title="크게" />
    </group>

    <group android:id="@+id/options_group" android:checkableBehavior="all">
        <item
            android:id="@+id/option_bold"
            android:title="굵게"
            android:checkable="true" />
        <item
            android:id="@+id/option_italic"
            android:title="기울임"
            android:checkable="true" />
    </group>
</menu>
```

예시 파일에서 〈group〉 태그의 android:checkableBehavior 속성은 아래와 같다.

- none: 그룹 내의 아이템들이 체크 가능하지 않습니다.
- single: 그룹 내에서 오직 하나의 아이템만 체크될 수 있습니다 (라디오 버튼처럼 동작).
- all: 그룹 내의 모든 아이템이 독립적으로 체크될 수 있습니다 (체크박스처럼 동작).

또 다른 메뉴 구성 방식 중에는 서브 메뉴(Sub Menu)라는 또 다른 형태가 있습니다. 서브 메뉴는 상위 메뉴 항목 아래에 하위 메뉴들을 표시하는 구조로, 사용자가 특정 메뉴를 선택했을 때 추가적인 선택지를 제공하는 방식입니다. 이를 구현하기 위해서는 하나의 메뉴 아이템(〈item〉) 안에 또 다른 〈menu〉 태그를 포함시켜 하위 메뉴를 정의하면 됩니다.

이 방식은 특히 기능을 단계적으로 제공하거나, 관련된 여러 옵션을 한 항목 아래에 모아두고

자 할 때 유용합니다.

예를 들어, "설정(Settings)" 메뉴를 클릭하면 화면 모드, 글꼴 크기, 알림 설정 등과 같은 세부 항목들이 하위 메뉴로 펼쳐지는 형태가 이에 해당합니다. 서브 메뉴를 사용하면 화면 공간을 절약하면서도 기능을 체계적으로 분류할 수 있어, 메뉴 구조를 보다 직관적이고 깔끔하게 유지할 수 있습니다.

아래는 서브 메뉴를 구성한 예시 화면과 해당 XML 파일의 코드로, 이를 참고하면 실제 앱 개발 시 서브 메뉴를 효율적으로 설계하고 구현하는 데 도움이 됩니다.또 다른 형태로 서브 메뉴 형태가 있다. 메뉴 아래 하위 메뉴를 표시하는 것을 말하며, 메뉴 아이템 내에 또 다른 〈menu〉 태그를 포함하여 서브 메뉴를 만들 수 있습니다. 아래는 서브 메뉴 구성 예시 화면과 파일입니다.

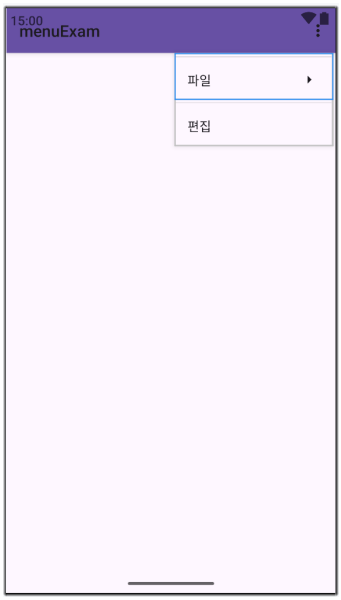


그림 4-2 서브 메뉴 구성 예시

메뉴 파일명.xml

```
<!-- 서브 메뉴 구성 예시 -->
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/file_menu"
        android:title="파일">
        <menu> <!-- 서브 메뉴 시작 -->
            <item
                android:id="@+id/new_file"
                android:title="새 파일" />
            <item
                android:id="@+id/open_file"
                android:title="열기" />
            <item
                android:id="@+id/save_file"
                android:title="저장" />
        </menu> <!-- 서브 메뉴 끝 -->
    </item>
    <item
        android:id="@+id/edit_menu"
        android:title="편집" />
</menu>
```

4.1.2 메뉴(Menu) 활성화

옵션 메뉴는 액티비티의 주 메뉴(Main Menu)로서, 일반적으로 화면 상단의 액션바(Action Bar) 또는 툴바(Toolbar)에 표시되며, 앱의 전반적인 기능이나 설정에 손쉽게 접근할 수 있도록 합니다. 이러한 옵션 메뉴를 활성화하려면, 먼저 이전 세션에서 다룬 것처럼 메뉴 리소스(XML 파일)를 구성해야 합니다. 이후, 액티비티 코드에서 메뉴를 표시하기 위한 메뉴 활성화 로직과, 각 메뉴 아이템을 클릭했을 때 실행할 이벤트 처리 코드를 작성하여야 합니다. 이를 통해 사용자는 메뉴를 통해 다양한 기능에 접근하고, 앱과 보다 직관적으로 상호작용할 수 있게 됩니다.

- 메뉴 활성화 : 액티비티의 onCreateOptionsMenu() 메서드를 오버라이드하여 메뉴 리소스 파일을 액티비티에 로드합니다. 아래는 활성화 코드 예시입니다.

```
// MainActivity.java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // MenuInflater를 사용하여 main_menu.xml 파일을 메뉴 객체로 변환
```

```
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true; // true를 반환하면 메뉴가 표시됩니다.
    }
```

- 메뉴 아이템 클릭 처리: 사용자가 옵션 메뉴의 아이템을 클릭했을 때의 동작을 정의하기 위해 onOptionsItemSelected() 메서드를 오버라이드합니다. 아래는 코드 예시입니다.

```
    // MainActivity.java
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId(); // 클릭된 메뉴 아이템의 ID를 가져와서 구분

        if (id == R.id.action_settings) {
            Toast.makeText(this, "설정 메뉴 클릭!", Toast.LENGTH_SHORT).show();
            return true; // 이벤트를 처리했음을 나타냅니다.
       } else if (id == R.id.action_search) {
            Toast.makeText(this, "검색 메뉴 클릭!", Toast.LENGTH_SHORT).show();
            return true;
        } else if (id == R.id.action_share) {
            Toast.makeText(this, "공유 메뉴 클릭!", Toast.LENGTH_SHORT).show();
            return true;
        } else if (id == R.id.action_exit) {
            Toast.makeText(this, "종료 메뉴 클릭!", Toast.LENGTH_SHORT).show();
            finish(); // 액티비티 종료
            return true;
        }

        // 처리되지 않은 이벤트는 부모 클래스에 전달
        return super.onOptionsItemSelected(item);
    }
```

(1) 액션바(ActionBar)와 툴바(Tool bar)

안드로이드에서 메뉴를 화면에 표시하는 방식은 크게 액션바(Action Bar)와 툴바(Toolbar) 두 가지 형태로 나눌 수 있습니다.

액션바(Action Bar)는 안드로이드 3.0(API 레벨 11)부터 도입된 표준 앱 바(App Bar)로, 액티비티의 테

마 설정에 따라 자동으로 제공되며, 앱 아이콘, 제목, 메뉴 아이템 등을 기본적으로 표시합니다. 다만, 액션바는 시스템이 기본 제공하는 구조를 사용하기 때문에 디자인 및 기능 커스터마이징에 한계가 있고, 최신 디자인 패턴 적용이나 하위 버전 기기와의 호환성 문제가 발생할 수 있습니다.

반면, 툴바(Toolbar)는 안드로이드 5.0(API 레벨 21)부터 도입된 보다 유연한 앱 바로, ViewGroup의 서브클래스이기 때문에 레이아웃(XML) 내부에서 자유롭게 배치할 수 있습니다. 또한, 색상, 높이, 버튼 배치, 글꼴 스타일 등 디자인과 기능 면에서 액션바보다 훨씬 더 폭넓은 커스터마이징이 가능합니다. 이러한 장점 덕분에 대부분의 최신 안드로이드 앱은 툴바를 사용하고, 이를 액션바로 설정하는 방식을 채택하고 있습니다.

툴바에 메뉴를 적용하려면, 먼저 레이아웃 편집 화면에서 [Palette] 〉 [Container] 〉 [Toolbar] 위젯을 선택하여 원하는 위치에 추가하면 됩니다. 이후 메뉴 리소스를 연결하고, 툴바를 액션바로 설정하는 코드를 작성하면 툴바를 활용한 메뉴 구성이 완성됩니다.

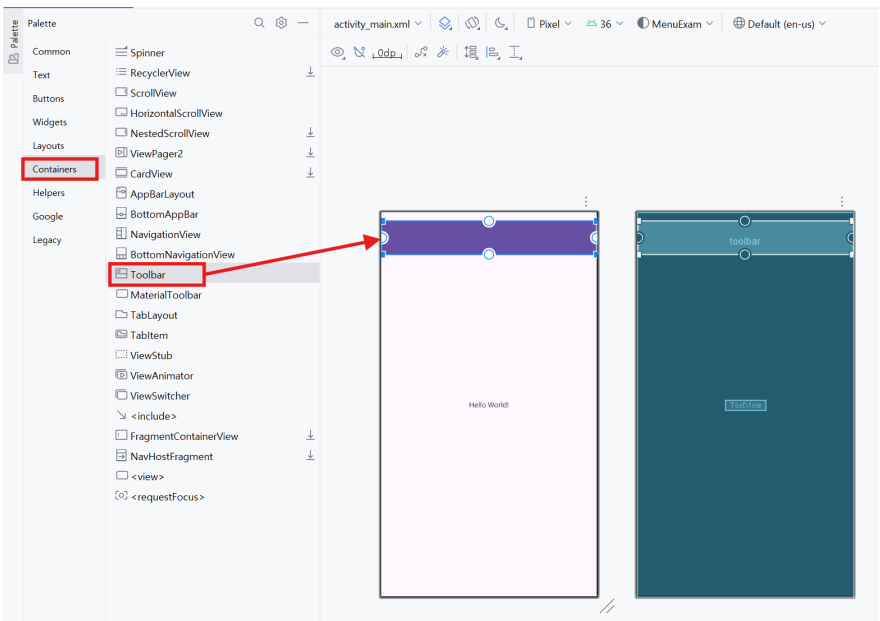


그림 4-3 Toolbar 위젯

이와 반대로, 기존의 액션바(Action Bar)에 메뉴를 설정하고자 한다면, 액티비티(Activity)에서 툴바(Toolbar)를 액션바로 지정해 주는 과정이 필요합니다. 즉, 툴바를 단순히 레이아웃에 배치하는 것만으로는 메뉴가 표시되지 않으며, 해당 툴바를 액션바로 인식하도록 코드에서 설정해야 합니다.

이를 위해 MainActivity.java의 onCreate() 메서드 안에서 findViewById()를 사용하여 레이아웃에 배치된 툴바 객체를 가져오고, setSupportActionBar() 메서드를 호출하여 해당 툴바를 액션바로 지정합니다. 이렇게 하면 툴바가 액션바 역할을 하게 되어, 이후 메뉴 리소스를 연결하거나 액션바의 다양한 기능(제목 표시, 네비게이션 버튼, 메뉴 아이템 등)을 활용할 수 있게 됩니다.

아래는 툴바를 액션바로 설정하는 예시 코드로, 이를 적용하면 기존 액션바를 대체하는 맞춤형 툴바를 사용할 수 있습니다.

```
// MainActivity.java
import androidx.appcompat.widget.Toolbar; // Toolbar 임포트

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

    Toolbar myToolbar = findViewById(R.id.my_toolbar);
    setSupportActionBar(myToolbar); // 툴바를 액션바로 설정
}
```

이제 onCreateOptionsMenu()와 onOptionsItemSelected() 메서드는 이 툴바에 자동으로 적용됩니다.

(2) 동적 메뉴 구성

안드로이드 앱에서는 특정 조건에 따라 메뉴의 동작과 상태를 실시간으로 변경할 수 있습니다. 예를 들어, 사용자의 로그인 여부나 현재 화면의 상태에 따라 메뉴 항목을 활성화 또는 비활성화할 수 있으며, 필요에 따라 숨기거나 표시하거나, 심지어 아이콘을 변경하는 것도 가능합니다.

이러한 동적 메뉴 구성을 구현할 때 유용하게 사용할 수 있는 메서드가 바로 onPrepareOptionsMenu()

입니다. 이 메서드는 옵션 메뉴가 화면에 표시되기 직전에 자동으로 호출되며, 이 시점에서 메뉴의 상태를 변경하면 사용자가 메뉴를 열었을 때 변경된 내용이 즉시 반영됩니다.

onPrepareOptionsMenu()를 오버라이드하면, 앱의 현재 상태나 특정 조건(예: 네트워크 연결 여부, 특정 데이터의 존재 여부, 사용자의 권한 수준 등)에 따라 메뉴 아이템을 동적으로 변경할 수 있습니다. 예를 들어, 어떤 메뉴 항목은 로그인이 되어 있을 때만 활성화되도록 하거나, 특정 기능이 사용할 수 없는 상태에서는 해당 메뉴를 숨기는 방식으로 구현할 수 있습니다. 또한, 상황에 맞춰 메뉴 아이콘을 다른 이미지로 교체하여 시각적인 변화를 줄 수도 있습니다.

아래에는 onPrepareOptionsMenu() 메서드를 활용하여 메뉴 아이템을 동적으로 제어하는 예시 코드가 제공되어 있으며, 이를 참고하면 보다 유연하고 사용자 친화적인 메뉴 구성이 가능합니다.

```
// MainActivity.java
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
    // 예시: 특정 조건에 따라 검색 메뉴를 활성화/비활성화
    MenuItem searchItem = menu.findItem(R.id.action_search);
    boolean someCondition = true; // 앱의 현재 상태에 따라 결정될 조건
    searchItem.setEnabled(someCondition); // 활성화 또는 비활성화
    searchItem.setVisible(someCondition); // 표시 또는 숨기기
    // 예시: 설정 메뉴 아이콘 변경
    if (someCondition) {
        menu.findItem(R.id.action_settings).setIcon(android.R.drawable.ic_menu_edit);
    } else {
        menu.findItem(R.id.action_settings).setIcon(android.R.drawable.ic_menu_preferences);
    }

    return super.onPrepareOptionsMenu(menu);
}
// 메뉴를 다시 그리도록 요청할 때
invalidateOptionsMenu(); // 이 메서드를 호출하면 onPrepareOptionsMenu()가 다시 호출됩니다.
```

(3) 검색 기능

SearchView는 안드로이드에서 제공하는 강력한 검색 기능 위젯으로, 액션바(Action Bar)에 직접 통합하여 사용할 수 있습니다. 이를 통해 사용자는 앱 화면을 벗어나지 않고도 바로 검색어를 입력하고, 실시간으로 검색 결과를 확인할 수 있습니다. 특히, 검색창이 메뉴 아이템 형태로

표시되기 때문에 화면 공간을 효율적으로 사용할 수 있으며, 앱 전반에 걸친 검색 기능 구현에 매우 유용합니다.

SearchView를 사용하기 위해서는 먼저 메뉴 XML 파일에 해당 위젯을 추가해야 합니다. 메뉴 XML에서 SearchView를 정의하면, 액션바의 메뉴 영역에 검색 아이콘이 표시되며, 사용자가 아이콘을 클릭하면 검색창이 확장되어 텍스트를 입력할 수 있는 상태가 됩니다. 이렇게 하면 검색 기능을 앱의 내비게이션 흐름과 자연스럽게 통합할 수 있습니다. 아래는 메뉴 XML에 SearchView를 추가하는 예시 코드로, 이를 기반으로 검색 기능을 액션바에 손쉽게 적용할 수 있습니다.

```
<item
    android:id="@+id/action_search_view"
    android:title="검색"
    android:icon="@android:drawable/ic_menu_search"
    app:showAsAction="collapseActionView|ifRoom"
    app:actionViewClass="androidx.appcompat.widget.SearchView" />
```

다음으로, 액티비티에서 활성화 되도록 아래와 코드를 작성해 줍니다.

```
// MainActivity.java
import androidx.appcompat.widget.SearchView; // SearchView 임포트

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu); // main_menu.xml 로드

    MenuItem searchItem = menu.findItem(R.id.action_search_view);
    SearchView searchView = (SearchView) searchItem.getActionView();

    if (searchView != null) {
        searchView.setQueryHint("검색어를 입력하세요...");
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                // 사용자가 검색 버튼을 눌렀을 때 또는 엔터를 쳤을 때
                Toast.makeText(MainActivity.this, "검색어 제출: " +
                                                    query, Toast.LENGTH_SHORT).show();
                searchView.clearFocus(); // 키보드 숨기기
                return true;
            }
```

```
            @Override
            public boolean onQueryTextChange(String newText) {
                // 검색어가 변경될 때마다 호출 (실시간 검색 기능 구현 시 활용)
                // Log.d("SearchView", "검색어 변경: " + newText);
                return false;
            }
        });
        searchView.setOnCloseListener(new SearchView.OnCloseListener() {
            @Override
            public boolean onClose() {
                // 검색 뷰가 닫힐 때
                Toast.makeText(MainActivity.this, "검색 뷰 닫힘",
                                                    Toast.LENGTH_SHORT).show();
                return false; // true 반환 시 닫히지 않음
            }
        });
    }
    return true;
}
```

(4) 옵션 메뉴 (Option Menu) 실습

Android API 33(Narwhal) 버전에서는 프로젝트 생성 시 기본 테마가 툴바(Toolbar)가 포함되지 않은 형태로 설정되어 있습니다. 따라서, 옵션 메뉴를 구현하고 실습하기 위해서는 툴바를 직접 추가하고, 메뉴 관련 리소스와 코드를 준비하는 과정이 필요합니다.

이번 실습에서는 옵션 메뉴 기능을 구현할 수 있도록 필요한 파일들을 생성하고 구성하는 단계를 순차적으로 진행하겠습니다. 구체적으로는 레이아웃 파일에 툴바를 배치하고, 메뉴 리소스(XML)를 작성하며, 액티비티 코드에서 툴바를 액션바로 설정한 뒤 메뉴를 연결하는 작업을 하게 됩니다. 이러한 준비 과정을 거치면, API 33 환경에서도 옵션 메뉴를 정상적으로 표시하고, 다양한 메뉴 이벤트 처리 기능을 실습할 수 있게 됩니다.

아래 제시된 절차에 따라 차근차근 진행해 주시기 바랍니다.

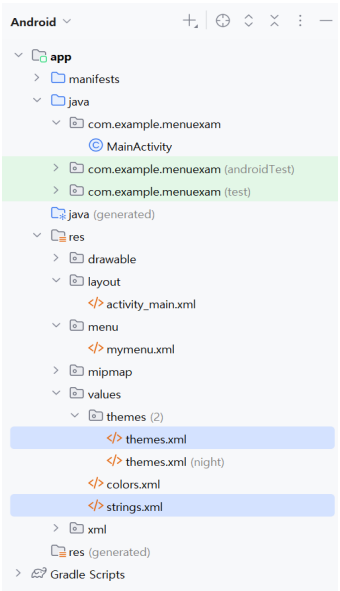


그림 4-4 themes.xml, strings.xml 위치

- string.xml 수정: app 〉 res 〉 values 폴더 내에 string.xml을 더블 클릭한 후 아래 코드를 입력해 주세요.

strings.xml

```
<resources>
    <string name="app_name">menuExam</string>
    <string name="menu_search">SEARCH</string>
    <string name="menu_help">HELP</string>
</resources>
```

- themes.xml 수정: app 〉 res 〉 values 〉 themes 폴더 내에 themes.xml 을 더블 클릭한 후 parent 속성값을 "Theme.Material3.DayNight" 으로 수정하세요.

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
<!-- Base application theme. -->
<style name="Base.Theme.MenuExam" parent="Theme.Material3.DayNight">
<!-- Customize your light theme here. -->
```

```
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
    </style>

    <style name="Theme.MenuExam" parent="Base.Theme.MenuExam" />
    </resources>
```

- mymenu.xml 생성: “4.1.1”에서 생성했던 menu 폴더 아래에 앞서 살펴본 내용을 참고하여 mymenu.xml 파일을 새로 만들었다면, 해당 파일을 아래와 같이 수정하세요. (“4.1.1”에서 해당파일을 생성하지 않았을 경우 생성)

mymenu.xml은 그래픽 편집기를 통해 시각적으로 구성할 수도 있지만, 이번 실습에서는 [Code] 모드를 활용하여 직접 코드를 입력하는 방식으로 진행하도록 하겠습니다.

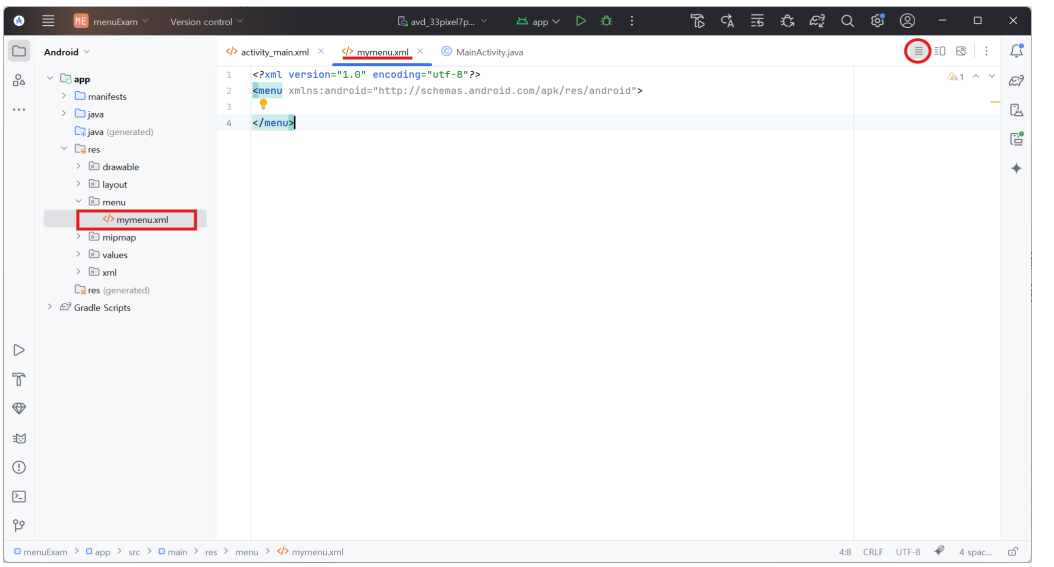


그림 4-5 menu 레이아웃 편집모드를 [Code] 모드로 변환

mymenu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/m_search"
        android:title="@string/menu_search"></item> <!-- strings.xml 참조 -->
    <item android:id="@+id/m_help"
        android:title="@string/menu_help"></item> <!-- strings.xml 참조 -->
</menu>
```

여기까지의 과정을 통해 메뉴 디자인 구성이 완료되었습니다. 이번 프로젝트에서는 레이아웃(layout) 구성은 프로젝트 생성 시 기본으로 제공되는 형태를 그대로 사용하고, 메인 액티비티(MainActivity.java)의 코드를 작성하여 메뉴 동작 여부를 확인하는 방식으로 실습을 진행하겠습니다. 따라서, 아래에 제시된 액티비티 코드를 작성한 뒤 프로젝트를 실행하여, 메뉴가 정상적으로 동작하는지 직접 확인해 보시기 바랍니다.

MainActivity.java

```
package com.example.menuexam;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
   MenuInflater inflater; // 메뉴XML 파일을 자바 코드로 불러오기 위한 객체 선언

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

   // 옵션 메뉴 생성 메서드(액티비티 실행 시 자동 호출)
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      super.onCreateOptionsMenu(menu);
      inflater = getMenuInflater();              // MenuInflater 객체 가져오기
      // res/menu/mymenu.xml에 정의된 메뉴를 메모리에 로드
      inflater.inflate(R.menu.mymenu, menu);
      return true; // 메뉴가 정상적으로 생성되었음을 알림
   }

   // 옵션 메뉴 아이템 선택 처리 메서드
   @Override
   public boolean onOptionsItemSelected(@NonNull MenuItem item) {
      int id = item.getItemId(); // 클릭된 메뉴 아이템의ID 가져오기
```

```
        // 메뉴ID에 따라 분기 처리
        if (id == R.id.m_search) { // "SEARCH" 메뉴 클릭 시
            Toast.makeText(this, "SEARCH 클릭", Toast.LENGTH_SHORT).show();
            return true;
        } else if (id == R.id.m_help) { // "HELP" 메뉴 클릭 시
            Toast.makeText(this, "HELP 클릭", Toast.LENGTH_SHORT).show();
            return true;
        }

        return false; // 처리되지 않은 메뉴 아이템은false 반환
    }
}
```

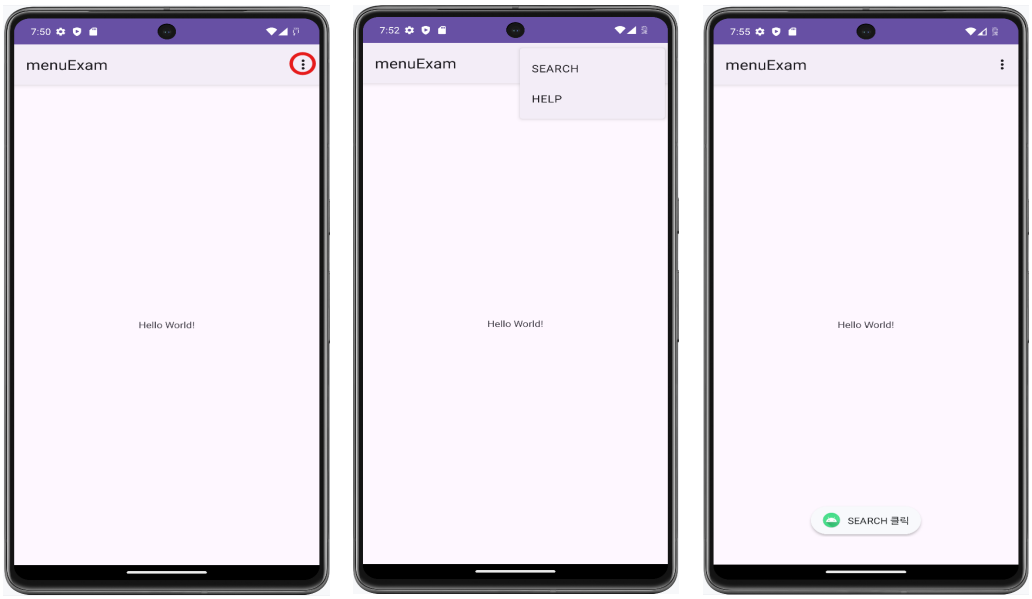


그림 4-6 옵션 메뉴 실행 화면

4.1.3 컨텍스트 메뉴(Context Menu)

컨텍스트 메뉴(Context Menu)는 안드로이드에서 제공하는 플로팅 메뉴(Floating Menu)의 한 형태로, 주로 특정 뷰(View) – 예를 들어 TextView, ImageView, 또는 ListView의 개별 아이템 – 를 길게 누르는(Long Click) 동작을 했을 때 화면에 표시됩니다. 이 메뉴는 사용자가 선택한 뷰와 직접적으로 연관된 작업을 제공하는 데 활용되며, 불필요한 기능을 노출하지 않고 상황에 맞는 기능만을 제공할 수 있다는 장점이 있습니다. 예를 들어, 텍스트를 길게 눌러 복사, 붙여넣기, 공

유 기능을 제공하거나, 이미지 뷰를 길게 눌러 저장 또는 삭제 기능을 제공하는 방식이 이에 해당합니다. 이번에는 컨텍스트 메뉴 구현 방법을 실습해 보기 위해, 아래에 제시된 절차에 따라 프로젝트를 설정하겠습니다. 이 과정에서는 레이아웃에 특정 뷰를 배치하고, 해당 뷰에 컨텍스트 메뉴를 등록한 뒤, 메뉴 선택 시 실행할 동작을 구현하게 됩니다.

- Project Name : contextExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) 컨텍스트 메뉴 생성 및 처리

옵션 메뉴 실습 프로젝트(menuExam)에서처럼 menu 폴더, mymenu.xml 파일을 구성하세요. 아래 스크립트 코드를 입력하기 위해서는 [Code] 모드로 변경해야 합니다.

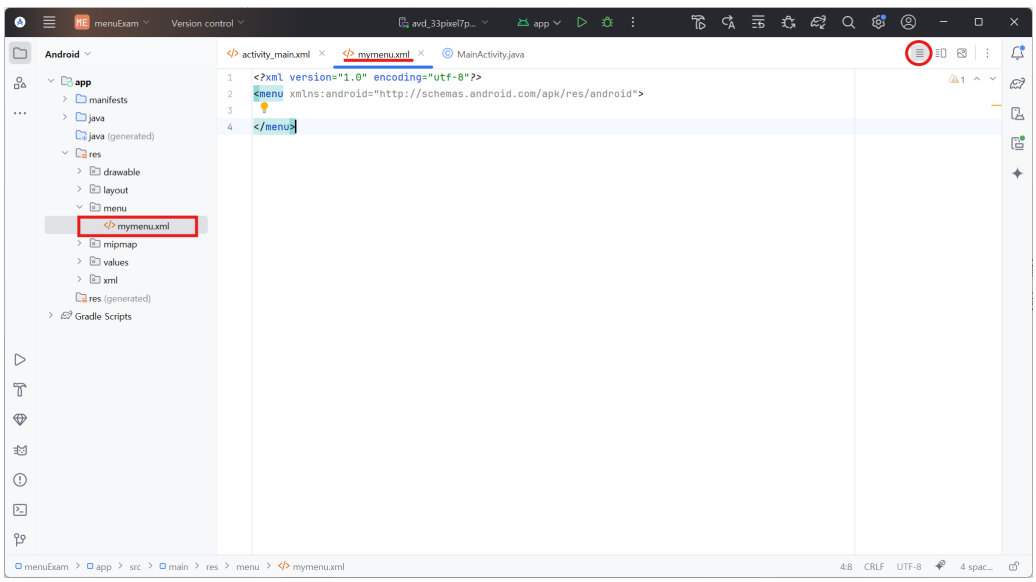
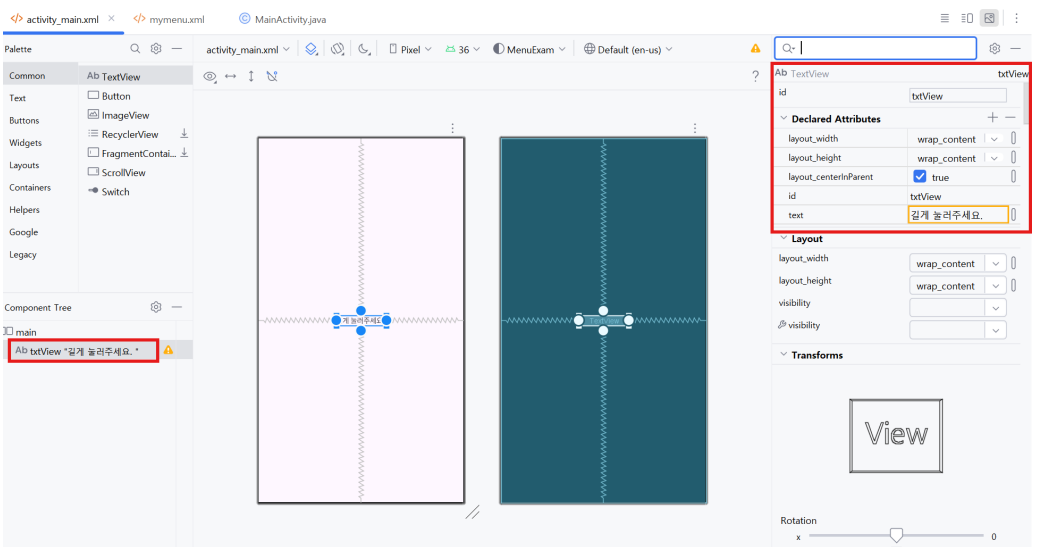


그림 4-7 menu 화면 [Code] 모드로 변환

mymenu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/context_edit"
          android:title="수정" />
    <item android:id="@+id/context_delete"
          android:title="삭제" />
</menu>
```

앞에서 설명한 것과 같이 컨텍스트(context) 메뉴는 뷰/위젯를 길게 눌렀을 때 나타나는 플로팅 메뉴이기 때문에 컨텍스트(context) 메뉴 실습을 위해서는 뷰/위젯이 필요하다. layout 설정 화면에서 RelativeLayout 으로 변경하고, 기본으로 추가 되어 있는 TextView 뷰/위젯의 id 속성을 "txtView", text 속성을 "길게 눌러주세요"로 변경하고, 다음으로 layout_centerInParent 속성을 "true"로 변경해 주세요.



레이아웃 및 뷰/위젯 설정을 완료한 후 아래 코드를 작성하세요.

MainActivity.java

```
package com.example.contextexam;

import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private TextView txtView; // 컨텍스트 메뉴를 표시할 대상TextView 선언

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

        // XML에서 정의한TextView와 자바 객체 연결
        txtView = findViewById(R.id.txtView);

        // 해당TextView를 컨텍스트 메뉴 대상 뷰(View)로 등록
        // → 사용자가TextView를 길게 누르면 컨텍스트 메뉴가 표시됨
        registerForContextMenu(txtView);
    }

    // 컨텍스트 메뉴 생성 메서드(TextView를 길게 눌렀을 때 호출됨)
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
                                    ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);

        // 메뉴를 띄울 대상 뷰 확인(여기서는txtView만 처리)
        if (v.getId() == R.id.txtView) {
            // res/menu/mymenu.xml 에 정의된 메뉴 리소스를 불러옴
            getMenuInflater().inflate(R.menu.mymenu, menu);

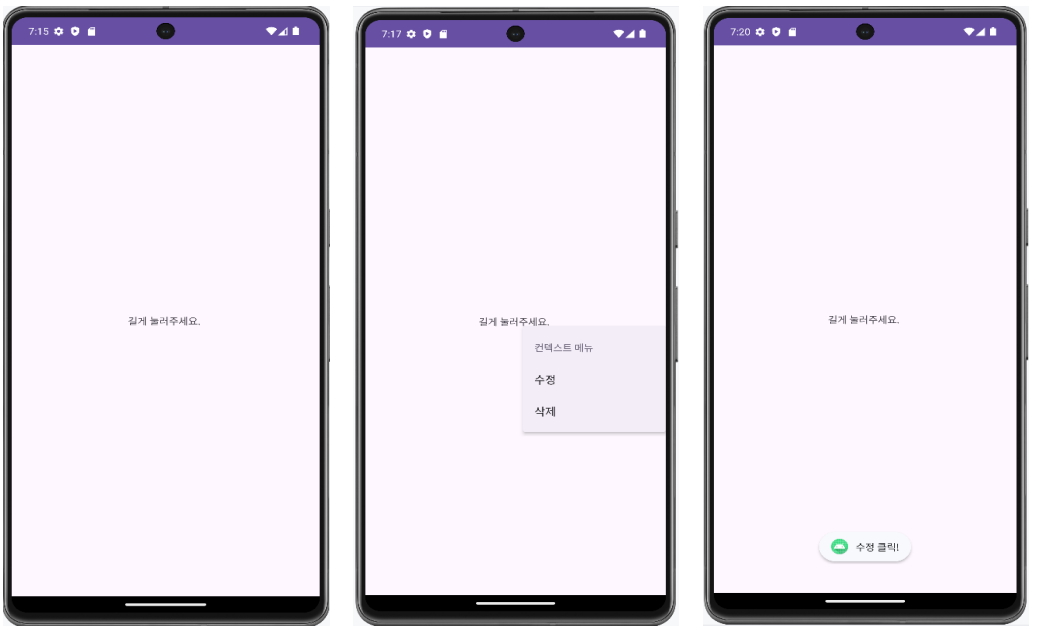
            // 메뉴 상단에 표시될 헤더 타이틀 지정
            menu.setHeaderTitle("컨텍스트 메뉴");
        }
    }

    // 컨텍스트 메뉴 항목 클릭 이벤트 처리 메서드
```

```
    @Override
    public boolean onContextItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId(); // 사용자가 선택한 메뉴 항목의ID 가져오기

        // ID에 따라 분기 처리
        if (id == R.id.context_edit) { // "수정" 메뉴 클릭 시
            Toast.makeText(this, "수정 클릭!", Toast.LENGTH_SHORT).show();
            return true;
        } else if (id == R.id.context_delete) { // "삭제" 메뉴 클릭 시
            Toast.makeText(this, "삭제 클릭!", Toast.LENGTH_SHORT).show();
            return true;
        }

        // 그 외 메뉴는 기본 동작 수행
        return super.onContextItemSelected(item);
    }
}
```



4.1.4 팝업 메뉴(Popup Menu)

팝업 메뉴(Popup Menu)는 안드로이드에서 제공하는 간단하고 직관적인 메뉴 형태로, 주로 특정 뷰(View) — 특히 Button과 같은 클릭 가능한 UI 요소 — 를 일반 클릭했을 때 해당 뷰 바로 아래에 짧게 표시됩니다. 이 메뉴는 화면 전체를 차지하지 않고, 선택된 뷰 주변에 작은 창 형태

로 나타나기 때문에 사용자가 현재 작업 흐름을 방해받지 않으면서 필요한 기능을 빠르게 선택할 수 있습니다.

팝업 메뉴는 컨텍스트 메뉴(Context Menu)와 비교했을 때 몇 가지 차이점이 있습니다. 컨텍스트 메뉴는 사용자가 뷰를 길게 눌러야 표시되지만, 팝업 메뉴는 길게 누르기(Long Click)가 아닌 일반 클릭(Single Click) 동작으로 즉시 표시됩니다. 이러한 특성 덕분에 팝업 메뉴는 특정 기능이나 작업에 대한 보조적인 선택지를 제공하는 데 자주 사용됩니다. 예를 들어, 목록의 각 항목 옆에 있는 버튼을 클릭했을 때 "편집", "삭제", "공유"와 같은 추가 옵션을 바로 보여주는 상황이 이에 해당합니다.

즉, 팝업 메뉴는 사용자의 작업 맥락(Context)에 맞는 선택지를 빠르게 제공하면서도 UI 흐름을 깔끔하게 유지할 수 있는 효율적인 인터페이스 요소라 할 수 있습니다.

(1) 팝업(Popup) 메뉴 생성 및 처리

Popup 메뉴 실습을 진행하기 위해, 아래에 제시된 절차에 따라 환경을 설정한 뒤 새로운 프로젝트를 생성하시기 바랍니다. 이렇게 준비 과정을 마친 후, 프로젝트에서 Popup 메뉴를 구현하고 동작을 직접 확인해 보겠습니다.

- Project Name : popupExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

다음 과정을 따라 실습을 진행해 보세요.

- 메뉴 리소스 파일 생성: popup_menu.xml 파일을 res/menu 폴더에 생성합니다.

앞서 구성했던 "menuExam" 프로젝트를 참고하여, 이번에는 popup_menu.xml 파일을 새로 생성한 뒤 아래에 제시된 스크립트를 작성해 주시기 바랍니다. 이렇게 하면 Popup 메뉴의 레이아웃과 항목 구성이 완료되어, 이후 액티비티 코드와 연동하여 동작을 구현할 수 있습니다.

popup_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/popup_item1"
         android:title="항목1" />
   <item android:id="@+id/popup_item2"
         android:title="항목2" />
   <item android:id="@+id/popup_item3"
         android:title="항목3" /></menu>
</menu>
```

- 팝업 메뉴 표시: layout을 RelativeLayout 으로 변경하고, 팝업 메뉴를 표시할 뷰(버튼)을 layout 설정 화면에서 추가하고, 아래와 같이 변경하세요. 그리고, 액티비티(MainActivity)에서 Button Component 클릭 리스너 내에서 PopupMenu 객체를 생성하고 show() 메서드를 호출합니다.

뷰/위젯	id	속성값	
button	btn1	layout_width	wrap_content
		layout_height	wrap_content
		layout_centerInParent	true
		text	PopUp

MainActivity.java

```
package com.example.popupexam;

import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.PopupMenu;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
```

```
    private Button popupMenuButton; // 팝업 메뉴를 띄울 버튼 변수 선언

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

        // XML에서 정의된 버튼(btn1)과 자바 객체 연결
        popupMenuButton = findViewById(R.id.btn1);

        // 버튼 클릭 이벤트 리스너 등록
        popupMenuButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // PopupMenu 객체 생성
                // 첫 번째 인자: 현재 컨텍스트(MainActivity)
                // 두 번째 인자: 팝업 메뉴가 붙을 기준 뷰(여기서는 버튼v)
                PopupMenu popup = new PopupMenu(MainActivity.this, v);

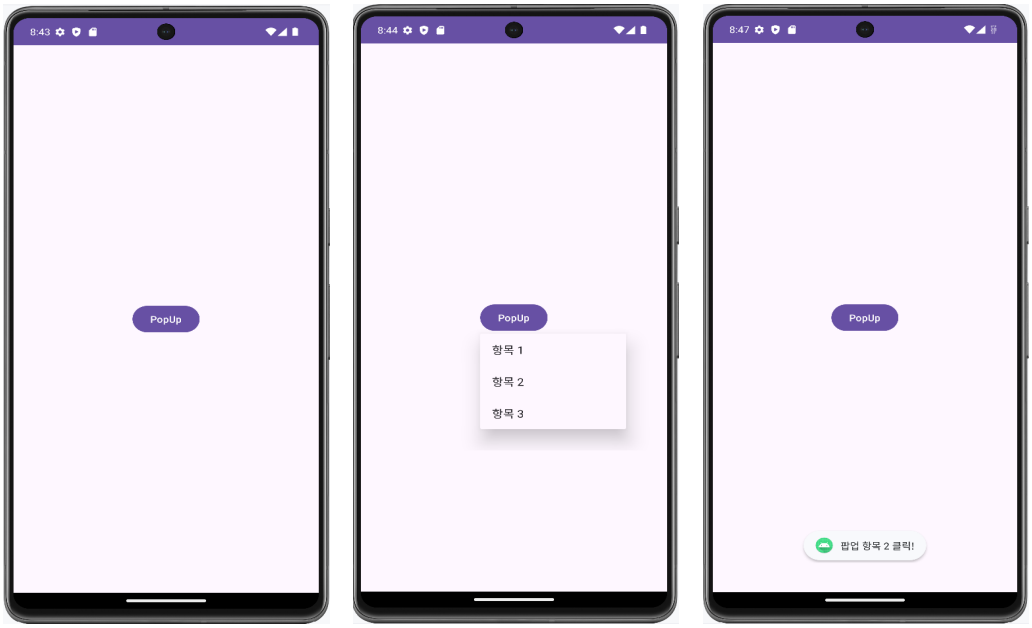
                // res/menu/popup_menu.xml 파일에 정의된 메뉴 리소스를 팝업 메뉴에 추가
                popup.getMenuInflater().inflate(R.menu.popup_menu, popup.getMenu());

                // 팝업 메뉴 항목 클릭 리스너 등록
                popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                    @Override
                    public boolean onMenuItemClick(MenuItem item) {
                        int id = item.getItemId(); // 사용자가 클릭한 메뉴 아이템의ID 확인

                        // 클릭한 아이템ID에 따라 분기 처리
                        if (id == R.id.popup_item1) {
                            Toast.makeText(MainActivity.this, "팝업 항목1 클릭!",
                                                                Toast.LENGTH_SHORT).show();
                            return true; // 이벤트 처리 완료
                        } else if (id == R.id.popup_item2) {
                            Toast.makeText(MainActivity.this, "팝업 항목2 클릭!",
                                                                Toast.LENGTH_SHORT).show();
                            return true;
                        } else if (id == R.id.popup_item3) {
                            Toast.makeText(MainActivity.this, "팝업 항목3 클릭!",
                                                                Toast.LENGTH_SHORT).show();
                            return true;
                        }
                        return false; // 처리되지 않은 경우false 반환
                    }
                });

                // 팝업 메뉴를 화면에 표시
```

```
                popup.show();
            }
        });
    }
}
```



(2) 팝업 메뉴의 스타일 변경

팝업 메뉴(Popup Menu)는 기본적으로 생성 시 지정한 앵커 뷰(View v)의 바로 아래 위치에 표시되도록 설계되어 있습니다. 즉, PopupMenu 객체를 만들 때 두 번째 인자로 전달하는 뷰가 기준점이 되어, 해당 뷰의 하단을 기준으로 메뉴가 나타나게 됩니다. 이 방식은 대부분의 일반적인 상황에서 직관적이고 편리하게 작동하기 때문에, 단순히 앵커 뷰를 지정하는 것만으로도 원하는 메뉴 표시 위치를 쉽게 구현할 수 있습니다.

하지만 경우에 따라서는 메뉴를 앵커 뷰의 기본 위치가 아닌 특정 위치에 표시해야 할 때가 있습니다. 예를 들어, UI 디자인 요구사항이나 사용자 경험(UX) 설계에 따라 메뉴를 화면 중앙, 특정 버튼 옆, 혹은 상단에 배치해야 하는 경우가 있습니다. 이런 경우에는 PopupMenu의 gravity 속성을 조정하거나, 앵커 뷰 자체의 레이아웃 위치를 변경하는 방식, 또는 팝업 표시 전 후처리 로직을 통해 좌표를 직접 계산하여 메뉴 위치를 세밀하게 제어하는 고급 기법을 사

용해야 합니다.

다만, 실무나 학습 초기 단계에서는 이러한 복잡한 위치 계산 대신, PopupMenu 객체를 생성할 때 앵커 뷰를 올바르게 지정하는 것만으로도 충분한 결과를 얻을 수 있습니다. 이를 실습에서 확인하기 위해, 이전에 작성했던 "popupExam" 프로젝트의 코드를 참고하여, 아래 예시와 같이 일부를 수정하면 팝업 메뉴의 위치와 동작 방식을 원하는 형태로 손쉽게 조정할 수 있습니다.

이 과정을 통해 기본적인 팝업 메뉴 사용법은 물론, 상황에 따른 메뉴 위치 제어 방법까지 익힐 수 있어, 향후 UI 구현 시 보다 유연하게 대응할 수 있을 것입니다.

```
popupMenuButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
          // PopupMenu 객체 생성 (컨텍스트, 팝업 메뉴가 붙을 뷰)
          PopupMenu popup = new PopupMenu(MainActivity.this, v, Gravity.END);
```

해당 코드 부분에 Gravity.END 를 추가한 뒤 실행하여 팝업 메뉴의 표시 위치 변화를 직접 확인해 보시기 바랍니다. 위치 설정에는 Gravity.START와 Gravity.END 값을 사용할 수 있으며, 이를 통해 팝업 메뉴를 앵커 뷰의 시작 위치 또는 끝 위치를 기준으로 표시할 수 있습니다.

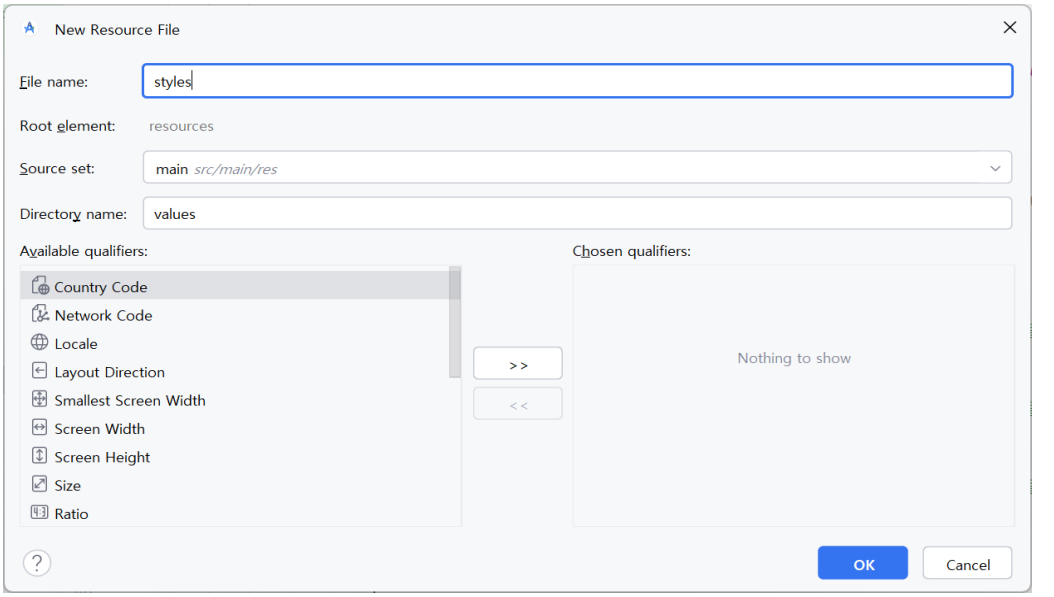
이제는 팝업 메뉴(Popup Menu)의 스타일을 변경하는 방법을 살펴보겠습니다. 팝업 메뉴의 스타일을 수정하려면, 앱의 테마 설정 파일(styles.xml)에서 PopupMenu 관련 속성을 오버라이드하는 방법과, 코드에서 popup.getMenu().add(...)를 사용해 메뉴 아이템을 추가하면서 개별 스타일을 지정하는 방법이 있습니다. 이 중 테마를 활용한 방식은 전체 팝업 메뉴의 스타일을 일괄적으로 변경할 수 있어, 유지보수와 일관성 측면에서 더 효율적입니다.

이를 위해 먼저 styles.xml 파일을 생성해야 합니다. 이 파일은 프로젝트의 app 〉 res 〉 values 폴더 안에 위치하며, 앱 전반의 스타일과 테마 속성을 정의하는 역할을 합니다. styles.xml 생성 방법은 이전에 "menuExam" 프로젝트에서 menu.xml 파일을 생성했던 과정과 매우 유사합니다. 즉, 프로젝트 네비게이터에서 해당 폴더를 열고 새 리소스 파일(Resource File)을 만들며, 리소스 종류(Resource Type)를 values로 지정한 후 파일 이름을 styles.xml로 입력하면 됩니다.

이 과정을 마치면, styles.xml 파일 안에서 팝업 메뉴와 관련된 속성을 오버라이드하여, 글꼴

크기, 배경 색상, 항목 간격, 아이콘 스타일 등 다양한 요소를 커스터마이징할 수 있습니다. 이렇게 구성한 스타일은 앱 전반에서 동일하게 적용되어, 팝업 메뉴의 시각적 일관성을 유지하면서도 브랜드나 디자인 가이드에 맞춘 UI를 구현할 수 있습니다.

- styles.xml 생성 : app 〉 res 〉 values 폴더에서 마우스 우클릭 후,
  new 〉 Values Resource File 선택하고, 파일명을 styles 로 입력



styles.xml 파일 생성후 해당 파일을 더블 클릭하여 아래 코드를 입력하면 됩니다.

styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="MyPopupMenu" parent="Widget.AppCompat.PopupMenu">
        <item name="android:popupBackground">@drawable/my_popup_background</item>
        <item name="android:textColor">#FF0000</item>
</style>
</resources>
```

- themes.xml 변경 :

app 〉 res 〉 values 〉 themes 폴더에 있으며, 다음과 같이 변경해 주세요.

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
<style name="Base.Theme.PopupExam" parent="Theme.Material3.DayNight.NoActionBar">
<item name="android:popupMenuStyle">@style/MyPopupMenu</item>
</style>

<style name="Theme.PopupExam" parent="Base.Theme.PopupExam" />
</resources>
```

• 팝업 메뉴 디자인 파일 생성 :

styles.xml 스크립트 내에서 사용된 @drawable/my_popup_background 는 팝업 메뉴의 배경 디자인 파일을 의미하며, 여기서 사용된 파일명은 예시를 위해 저자가 임의로 지정한 것이므로 참고시 이 점을 유념하시기 바랍니다.

– 생성 방법: 프로젝트 탐색기에서 app 〉 res 〉 drawable 폴더를 마우스 우측 버튼으로 클릭한 뒤 New 〉 Drawable Resource File을 선택합니다.

파일 이름은 my_popup_background 로 지정합니다.

생성된 @drawable/my_popup_backgroud 파일을 더블 클릭하여 코드 작성화면에서 아래와 같이 입력해 주세요.

my_popup_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle">
    <!-- 배경 색상 -->
    <solid android:color="#E4F7BA" />
    <!-- 모서리 둥글게 -->
    <corners android:radius="8dp" /> <!-- 8dp 둥근 모서리 -->
    <!-- 테두리 선 -->
    <stroke android:width="1dp" android:color="#CCCCCC" />

    <padding
        android:bottom="2dp"
        android:left="2dp"
        android:right="2dp"
        android:top="2dp" />
</shape>
```

모든 입력을 마무리 한 후 실행하여 팝업 메뉴 디자인과 팝업 위치 변경이 이루어졌는지 확인해 보시기 바랍니다.

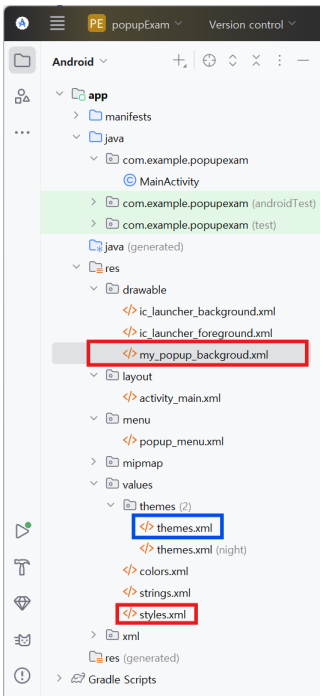


그림 4-8 팝업메뉴 스타일 변경을 위한 파일 목록 및 위치

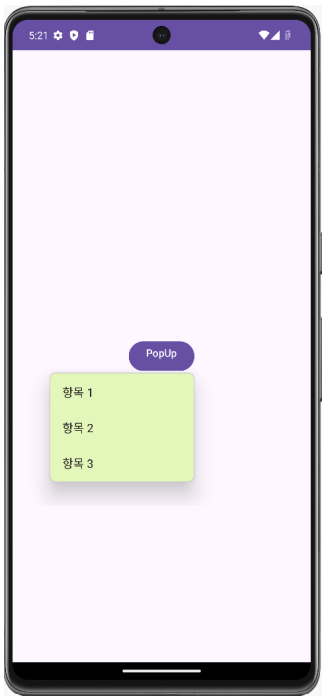


그림 4-9 스타일 변경된 팝업 메뉴 실행화면

4.2 대화상자(Dialog)

대화상자는 안드로이드 앱에서 사용자의 주의를 집중시키고 특정 행동을 유도하기 위해 표시되는 작은 팝업 창으로, 보통 중요한 정보를 전달하거나, 사용자의 결정을 요구하거나, 간단한 입력을 받을 때 활용됩니다. 대화상자는 화면 위에 별도의 창으로 나타나므로, 사용자가 현재 진행 중인 작업의 흐름을 잠시 멈추고 대화상자에서 제시하는 메시지나 옵션에 반응하도록 합니다. 이러한 특성 때문에 대화상자는 사용자와 앱 간의 상호작용에서 매우 중요한 역할을 하며, 올바르게 설계하면 앱의 사용성을 크게 향상시킬 수 있습니다.

안드로이드에서 일반적으로 많이 사용되는 대화상자 형태에는 몇 가지 대표적인 예가 있습니다. 가장 기본적인 형태인 AlertDialog(알림 대화상자)는 메시지를 사용자에게 전달하고, 확

인(OK), 취소(Cancel)와 같은 버튼을 통해 간단한 결정을 받는 데 사용됩니다. 또 다른 예로는 DatePickerDialog(날짜 선택 대화상자)가 있는데, 이는 달력 UI를 통해 사용자가 원하는 날짜를 직관적으로 선택할 수 있도록 도와줍니다. 이와 유사하게, TimePickerDialog(시간 선택 대화상자)는 시계 UI를 제공하여 사용자가 특정 시간 값을 쉽게 입력할 수 있도록 합니다. 즉, 대화상자는 단순한 정보 전달부터 사용자 입력에 이르기까지 다양한 용도로 사용되며, 상황에 따라 적절한 형태의 대화상자를 선택하고 구현하는 것이 앱의 사용자 경험(UX)을 높이는 핵심 요소라고 할 수 있습니다.

4.2.1 AlertDialog (경고 대화상자)

AlertDialog는 안드로이드에서 가장 일반적으로 사용되는 대화상자 유형으로, 사용자에게 중요한 메시지를 직관적으로 전달하고 동시에 하나 이상의 버튼을 제공하여 사용자의 응답이나 선택을 받을 수 있는 기능을 갖추고 있습니다. 이 대화상자는 단순히 알림만을 표시하는 것이 아니라, 긍정(Positive), 부정(Negative), 그리고 필요하다면 중립(Neutral) 버튼까지 제공하여 상황에 맞는 다양한 선택지를 사용자에게 제시할 수 있습니다.

예를 들어, "파일을 삭제하시겠습니까?"라는 메시지를 띄운 뒤, 긍정 버튼에는 "삭제", 부정 버튼에는 "취소"와 같은 동작을 지정할 수 있습니다. 또 다른 경우에는 "앱을 평가하시겠습니까?"라는 메시지를 띄우고, 긍정 버튼은 "예", 부정 버튼은 "아니오", 중립 버튼은 "나중에"로 설정하여 사용자가 보다 다양한 방식으로 응답할 수 있도록 구성할 수도 있습니다.

이처럼 AlertDialog는 사용자와 앱 간의 즉각적이고 명확한 상호작용을 가능하게 하며, 사용자에게 중요한 결정을 요구하거나 알림을 제공하는 데 매우 효과적인 UI 요소입니다.

(1) AlertDialog 생성 및 표시

AlertDialog 실습을 위해 아래와 같이 설정하고 새로운 Project를 생성하세요.

- Project Name : alertExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

이번에 진행할 실습 프로젝트는 웹사이트 접속 기능을 포함하고 있기 때문에, 이에 필요한 권한 설정 또한 함께 다루도록 하겠습니다. 안드로이드 앱에서 외부 웹사이트에 접속하기 위해서는 반드시 애플리케이션에 인터넷 사용 권한을 부여해야 합니다.

이를 위해 위 설정대로 프로젝트를 생성한 뒤, Project Navigator(View)를 살펴보면 app 〉 manifest 폴더를 확인할 수 있을 것입니다. 해당 폴더에 AndroidManifest.xml 파일을 선택한 후 해당 파일을 더블 클릭하여 열고, 아래에 제시된 코드를 추가해 주시면 됩니다. 이 과정을 통해 앱이 외부 네트워크와 통신할 수 있는 권한을 얻게 되며, 웹사이트 접속 기능이 정상적으로 동작 시킬 수 있습니다.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools">

    <!-- 해당 부분 추가 -->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules“
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AlertExam">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

다음으로 layout 구성화면에서 Relativelayout 으로 변경하고, 아래 표와 같이 뷰/위젯(component)를 추가하세요.

component	속성값	
button	layout_width	wrap_content
	layout_height	wrap_content
	layout_marginStart	30dp
	layout_marginTop	30dp
	id	btn_naver
	text	Naver
button	layout_width	wrap_content
	layout_height	wrap_content
	layout_marginStart	270dp
	layout_marginTop	30dp
	id	btn_google
	text	Google

component	속성값	
button	layout_width	wrap_content
	layout_height	wrap_content
	layout_marginStart	140dp
	layout_marginTop	30dp
	id	btn_daum
	text	Daum

다음으로, AlertDialog.Builder 클래스를 사용하여 대화상자를 단계적으로 구성하기 위해 액티비티 내 아래 코드를 입력하세요.

MainActivity.java

```
package com.example.alertexam;

import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;


// MainActivity 클래스: AppCompatActivity를 상속받고, 버튼 클릭 이벤트 처리를 위해
OnClickListener 인터페이스 구현
public class MainActivity extends AppCompatActivity implements
```

```
View.OnClickListener {

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

   // 레이아웃에 정의된 버튼 객체 가져오기
   Button btn_n = (Button) findViewById(R.id.btn_naver);
   Button btn_d = (Button) findViewById(R.id.btn_daum);
   Button btn_g = (Button) findViewById(R.id.btn_google);

   // 현재 액티비티를 클릭 리스너로 등록
   btn_n.setOnClickListener(this);
   btn_d.setOnClickListener(this);
   btn_g.setOnClickListener(this);
}

@Override
public void onClick(View v) {
   int id = v.getId(); // 클릭된 버튼의id를 가져와 구분
   if (id == R.id.btn_naver)
      dialog_run("Naver", Uri.parse("http://m.naver.com"));
   else if(id == R.id.btn_daum)
      dialog_run("Daum", Uri.parse("http://m.daum.net"));
   else if(id == R.id.btn_google)
      dialog_run("Google", Uri.parse("http://www.google.com"));
}

// AlertDialog 실행 메서드
public void dialog_run(String str, final Uri uri) {
   AlertDialog.Builder dialogbuilder = new AlertDialog.Builder(this);

   // 대화상자 제목 설정
   dialogbuilder.setTitle("알림");

   // 대화상자 메시지 설정
   dialogbuilder.setMessage(str + "로 이동하시겠습니까?");

   // 긍정(예) 버튼 설정: 누르면 해당uri로 이동
   dialogbuilder.setPositiveButton("예", new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
         Intent it = new Intent(Intent.ACTION_VIEW, uri); // 웹 브라우저 열기
         startActivity(it);
      }
   });
```

```
        // 부정(아니오) 버튼 설정: 누르면 토스트 메시지 표시
        dialogbuilder.setNegativeButton("아니오", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getApplicationContext(), "이동을 취소하셨습니다.",
                                        Toast.LENGTH_LONG).show();
            }
        });

        // 대화상자 생성 및 표시
        AlertDialog dialog = dialogbuilder.create();
        dialog.show();
    }
}
```



코드 설명

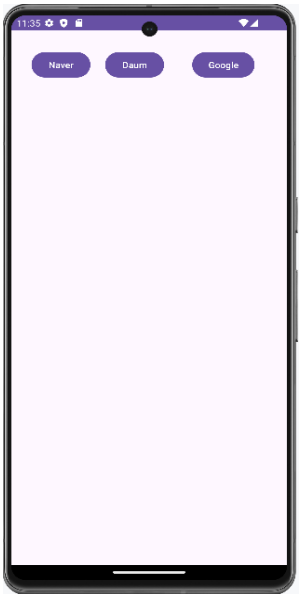
- public void onClick(View v) { .....
  - 세 개의 버튼 중 어떤 버튼이 눌러졌는지 확인하기 위해 v.getId()로 확인
  - 버튼 ID(R.id.btn_naver, R.id.btn_daum, R.id.google)에 따라 사용자가 정의한 dialog_run() 메서드를 호출하며, 사이트 명칭과 접속할 URL을 넘겨줍니다.

- public void dialog_run(String str, final Uri uri) { .....

  → `dialogbuilder.setPositiveButton("예", new DialogInterface.OnClickListener() {`

```
    @Override
    public void onClick(DialogInterface dialog, int which) {
        Intent it = new Intent(Intent.ACTION_VIEW, uri);
        startActivity(it);
    }
}); :
```

  - 긍정버튼 ("예") 클릭시 실행될 코드 정의
  - public void onClick(View v) 메서드에서 사용자가 클릭한 버튼 ID에 따라 해당 버튼에 매핑된 URL값을 파라메터 uri가 전달받고, uri를 이용해 Intent.ACTION_VIEW 동작을 실행
  - Intent.ACTION_VIEW 는 넘겨받은 데이터(uri) 형태에 맞는 안드로이드 기기 내 설치된 앱 중 기본 설정 앱으로 실행하라는 의미이며, 여기서는 넘겨받은 데이터가 URL 형태이므로 웹브라우저 를 실행







주요 개념

AlertDialog 클래스는 실제로 화면에 표시되는 다이얼로그 객체이지만, 이 클래스의 생성자는 protected로 선언되어 있기 때문에 개발자가 직접 new AlertDialog()를 호출하여 객체를 생성할 수는 없습니다. 따라서 다이얼로그를 만들기 위해서는 반드시 AlertDialog.Builder 클래스를 활용해야 합니다.

AlertDialog.Builder는 다이얼로그를 구성하기 위한 다양한 설정 기능을 제공하며, 이를 통해 다이얼로그의 제목(Title), 메시지(Message), 버튼(Button) 등 필요한 요소들을 미리 정의할 수 있습니다. 개발자는 이 Builder 객체에 원하는 속성을 차례대로 설정한 뒤, 최종적으로 create() 또는 show() 메서드를 호출하여 실제 AlertDialog 객체를 생성하고 화면에 표시하게 됩니다.

즉, 안드로이드에서 AlertDialog를 실행하기 위해서는 단순히 객체를 직접 생성하는 방식이 아니라, Builder 패턴을 통해 필요한 속성을 먼저 설정하고, 그 결과를 기반으로 다이얼로그 객체를 얻어내는 구조를 반드시 거쳐야만 합니다. 이러한 방식은 다이얼로그의 유연한 커스터마이징을 가능하게 하고, 코드의 가독성과 유지보수성 또한 향상 시켜 줍니다.

(2) 목록형 대화상자

앞에서 살펴본 기본적인 형태의 AlertDialog는 단순히 메시지를 표시하고 사용자의 확인이나 취소와 같은 단순 응답을 받는 방식이었습니다. 그러나 실제 앱 개발에서는 경우에 따라 사용자가 여러 항목 중 하나를 선택해야 하는 상황이 자주 발생합니다. 이럴 때는 목록형 AlertDialog를 활용하면 사용자가 선택 가능한 항목들을 리스트 형태로 제공하고, 그 중 원하는 항목을 직접 고를 수 있도록 구현할 수 있습니다.

이번에는 이러한 목록형 AlertDialog의 구현 방식을 실습해 보겠습니다. 이를 위해 새로운 프로젝트를 생성하고, 아래에 제시된 단계에 따라 환경을 설정한 뒤 코드를 작성해 보시기 바랍니다. 이렇게 하면 기본 다이얼로그와 달리, 선택 가능한 리스트가 포함된 대화상자를 생성하고 동작을 확인할 수 있습니다.

목록형 AlertDialog 실습을 위해 위해 아래와 같이 설정하고 새로운 Project를 생성하세요.

- Project Name : alertlist
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

프로젝트 "alertExam" 설정했던 것처럼 Androidmanifest.xml 에 네트워크를 사용할 수 있도록 권한을 설정하세요.

레이아웃(layout) 설정은 RelativeLayout 으로 하고, component 설정은 아래와 같습니다.

component	속성값	
button	layout_width	wrap_content
	layout_height	wrap_content
	layout_centerInParent	true
	id	btn_web
	text	버튼을 누르세요.

화면 구성 완료 후 아래와 같이 코드를 입력하세요.

MainActivity.java

```
package com.example.alertlist;

import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    int select = -1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button btn_select = (Button) findViewById(R.id.btn_web);
        btn_select.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        // 대화상자에 표시될 아이템 목록(배열로 정의)
        final String[] items = {"Naver", "Daum", "Google"};
        // 아이템 목록에 대응하는 목록으로 대화상자에서 선택시 이동할url (배열로 정의)
        final String[] url =
                    {"http://m.naver.com", "http://m.daum.net", "http://www.google.com"};

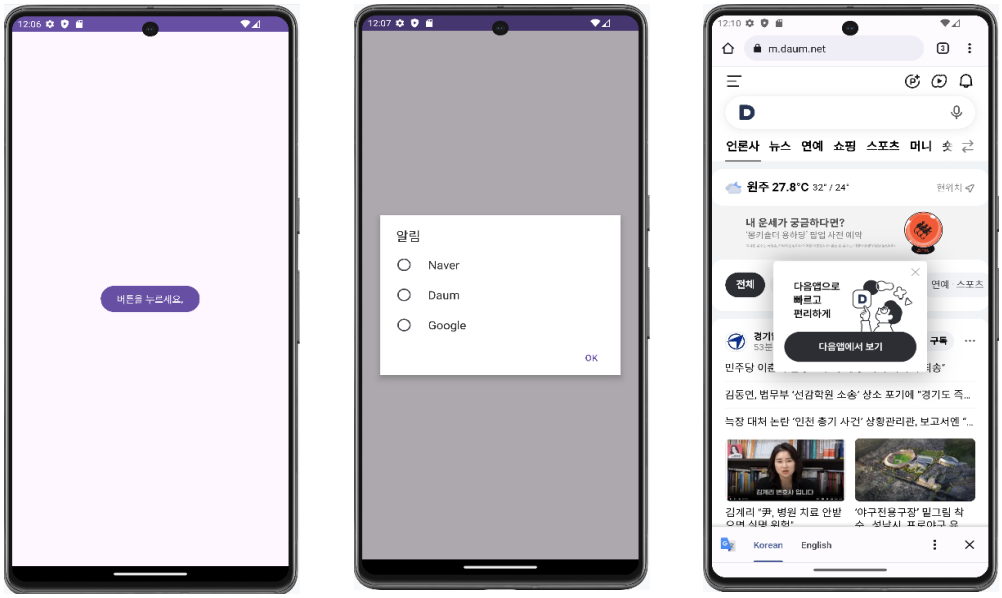
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("알림");

        // 단일 선택형 다이얼로그 박스: setSingleChoiceItems
        // 다중 선택형 다이얼로그 박스: setMultiChoiceitems
        builder.setSingleChoiceItems(items, -1, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // 목록 중 선택한 아이템의 위치값(which)을select에 반환
```

```
                select = which;
            }
        });

        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Intent it = new Intent(Intent.ACTION_VIEW, Uri.parse(url[select]));
                startActivity(it);
            }
        });

        AlertDialog dialog = builder.create();
        dialog.show();
    }
}
```



(3) 사용자 정의(custom) 대화상자

AlertDialog는 builder.setView(View view) 메서드를 사용하여 완전히 사용자 정의된 레이아웃을 대화상자 내용으로 포함할 수 있습니다. 이를 통해 로그인 폼, 복잡한 설정 화면 등 다양한 형태의 대화상자를 만들 수 있습니다.

사용자 정의 AlertDialog 실습을 위해 아래와 같이 설정하고 새로운 Project를 생성하세요.

- Project Name : customExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

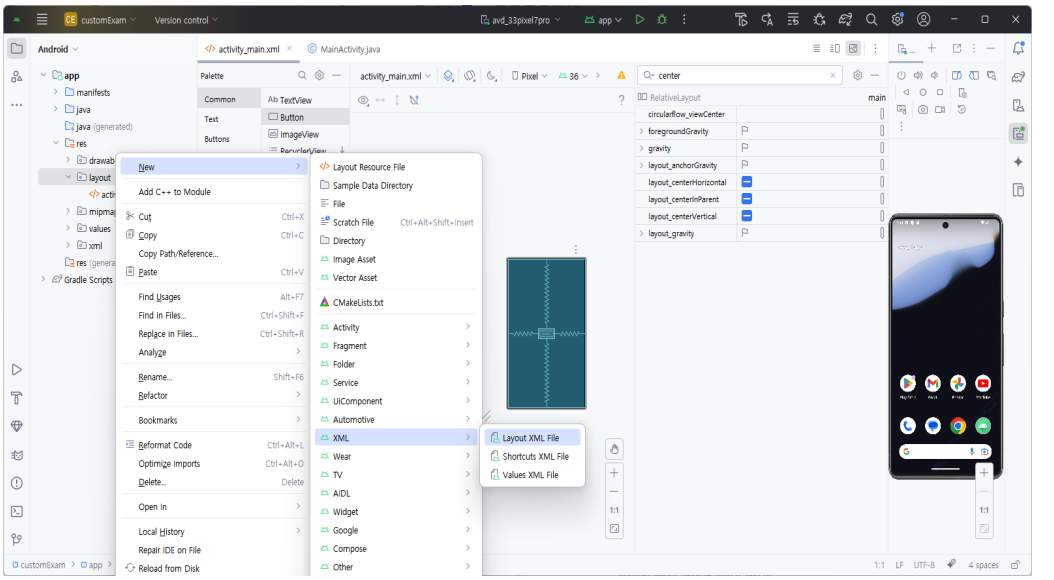
레이아웃(layout) 설정은 RelativeLayout 으로 하고, component 설정은 아래와 같습니다.

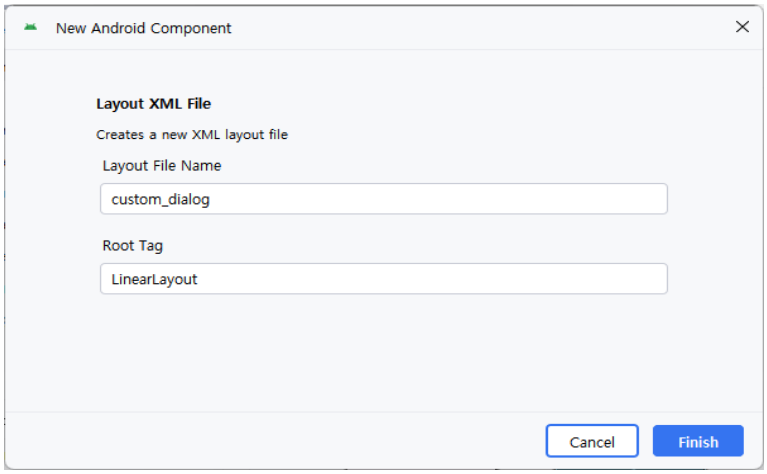
component	속성값	
button	layout_width	wrap_content
	layout_height	wrap_content
	layout_centerInParent	true
	id	btn_login
	text	로그인

다음으로, 커스텀 대화상자를 위한 레이아웃(layout) 생성을 위해 아래 과정을 진행해 주세요.

app 〉 res 〉 layout 폴더를 마우스 우측 클릭 후 New 〉 XML 〉 Layout XML File 선택하세요.

Layout File명은 custom_dialog로 입력하세요.





커스텀 다이얼로그 디자인이 다소 복잡한 관계로 [Code] 모드로 변환하여 아래 코드를 입력하세요.

custom_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="20dp">

    <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="로그인"
      android:textSize="22sp"
      android:textStyle="bold"
      android:layout_gravity="center_horizontal"
      android:layout_marginBottom="20dp" />

    <EditText
      android:id="@+id/usernameEditText"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="아이디"
      android:inputType="text"
      android:layout_marginBottom="10dp" />

    <EditText
```

```
        android:id="@+id/passwordEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="비밀번호"
        android:inputType="textPassword"
        android:layout_marginBottom="20dp" />

    <Button
        android:id="@+id/loginButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="로그인" />

</LinearLayout>
```

custom_dialog.xml 완성 후, 아래와 같이 액티비티(MainActivity.java) 파일을 완성하세요.

MainActivity.java

```
package com.example.customexam;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

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

        Button btn_in = (Button) findViewById(R.id.btn_login);
        btn_in.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
        // 커스텀 레이아웃 인플레이트
        View customLayout = getLayoutInflater().inflate(R.layout.custom_dialog, null);
        builder.setView(customLayout); // 대화상자에 커스텀 뷰 설정
```

```
        builder.setCancelable(false); // 외부 터치로 닫히지 않도록 설정

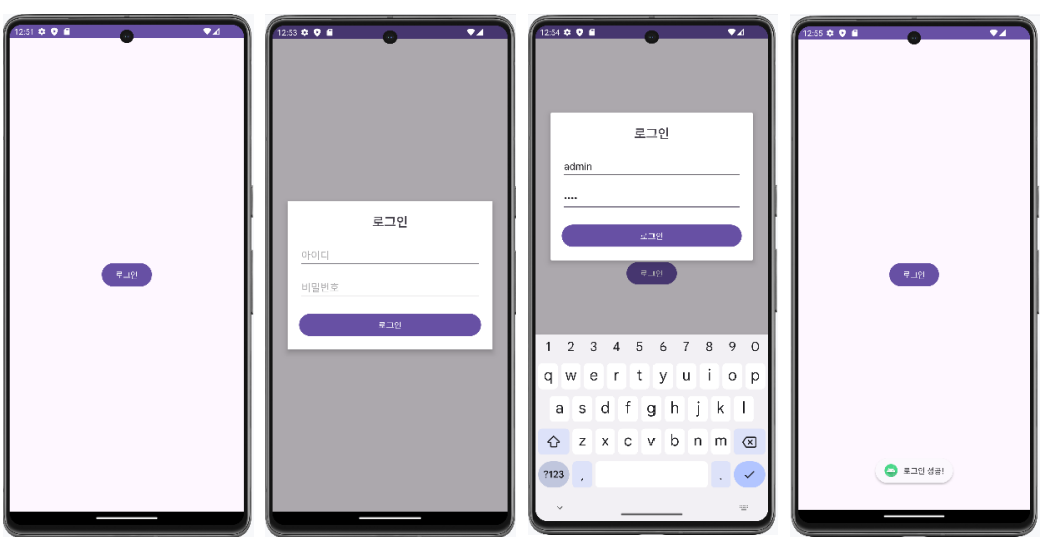
        final EditText usernameEditText = customLayout.findViewById(R.id.usernameEditText);
        final EditText passwordEditText = customLayout.findViewById(R.id.passwordEditText);
        Button loginButton = customLayout.findViewById(R.id.loginButton);

        final AlertDialog dialog = builder.create(); // 대화상자 생성

        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = usernameEditText.getText().toString();
                String password = passwordEditText.getText().toString();

                if (username.equals("admin") && password.equals("1234")) {
                    Toast.makeText(MainActivity.this, "로그인 성공!",
                                                            Toast.LENGTH_SHORT).show();
                    dialog.dismiss(); // 대화상자 닫기
                } else {
                    Toast.makeText(MainActivity.this, "아이디 또는 비밀번호가 틀렸습니다.",

                                                            Toast.LENGTH_SHORT).show();
                }
            }
        });
        dialog.show();
    }
}
```



4.2.3 날짜와 시간 Dialog (DatePickerDialog, TimePickerDialog)

날짜와 시간을 선택할 수 있도록 해주는 DatePickerDialog와 TimePickerDialog는 안드로이드에서 유용하게 사용할 수 있는 도구이다.

DatePickerDialog 다이얼로그는 날짜이고, TimePickerDialog는 시간을 선택할 수 있는 다이얼로그이다.

실습을 위해 아래 설정으로 Project를 생성하세요.

- Project Name : datetimeExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

레이아웃(layout)은 [Code] 모드로 변환하여 아래 코드를 입력하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

   <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginLeft="10dp"
      android:layout_marginTop="40dp"
      android:layout_marginRight="10dp"
      android:orientation="horizontal" >

   <TextView
      android:id="@+id/txt_date"
      android:layout_width="80dp"
      android:layout_height="25dp"
      android:layout_gravity="center_vertical"
      android:layout_weight="3" />

    <Button
      android:id="@+id/btn_date"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
        android:layout_marginLeft="20dp"
        android:layout_weight="1"
        android:text="날짜 설정" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="5dp"
        android:layout_marginRight="10dp"
        android:orientation="horizontal">

    <TextView
        android:id="@+id/txt_time"
        android:layout_width="80dp"
        android:layout_height="25dp"
        android:layout_gravity="center_vertical"
        android:layout_weight="3" />

    <Button
        android:id="@+id/btn_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_weight="1"
        android:text="시간 설정" />

     </LinearLayout>
</LinearLayout>
```

화면 구성 및 component 설정이 완료되면 아래의 코드를 입력하세요.

MainActivity.java

```
package com.example.datetimeexam;

import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;

import androidx.appcompat.app.AppCompatActivity;
```

```
import java.util.Calendar;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    // 버튼과 텍스트뷰 선언
    Button btn_selDate, btn_selTime;
    TextView tv_Date, tv_Time;

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

        // XML 레이아웃에서 버튼과 텍스트뷰 객체 찾기
        btn_selDate = (Button) findViewById(R.id.btn_date);
        btn_selTime = (Button) findViewById(R.id.btn_time);

        // 버튼 클릭 리스너 등록(현재 액티비티를 리스너로 지정)
        btn_selDate.setOnClickListener(this);
        btn_selTime.setOnClickListener(this);

        // 날짜/시간을 출력할 텍스트뷰 객체 찾기
        tv_Date = (TextView) findViewById(R.id.txt_date);
        tv_Time = (TextView) findViewById(R.id.txt_time);
    }

    @Override
    public void onClick(View v) {
        // 클릭된 뷰의ID 확인
        int id = v.getId();

        // 날짜 선택 버튼 클릭 시dateSet() 실행
        if (id == R.id.btn_date) {
            dateSet();
        } else if(id == R.id.btn_time) { // 시간 선택 버튼 클릭 시timeSet() 실행
            timeSet();
        }
    }

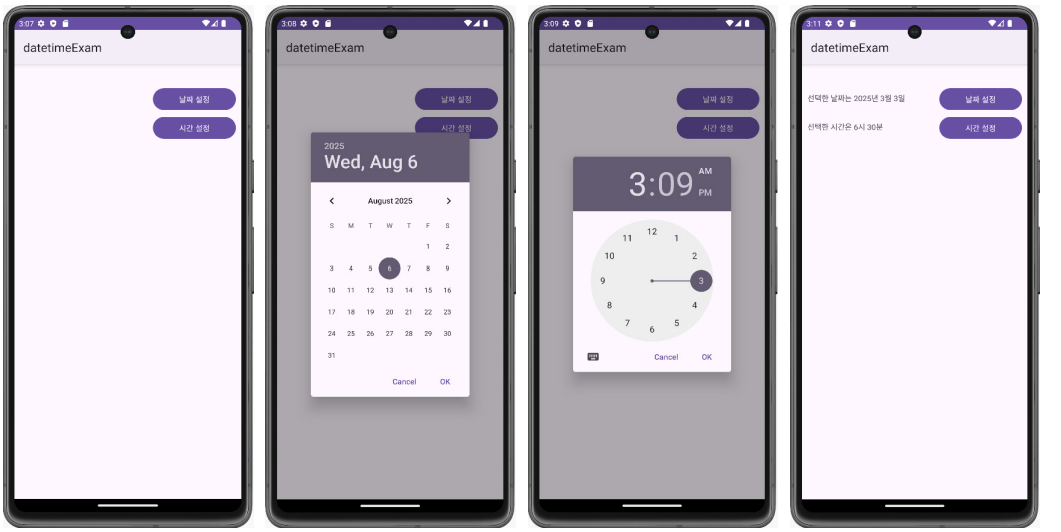
    // 날짜 설정 다이얼로그를 띄우는 사용자 정의 메소드
    public void dateSet() {
        final Calendar cd = Calendar.getInstance();   // 현재 날짜/시간 가져오기
        int mYear = cd.get(Calendar.YEAR);            // 연도
        int mMonth = cd.get(Calendar.MONTH);          // 월(0부터 시작→ +1 필요)
        int mDay = cd.get(Calendar.DAY_OF_MONTH);     // 일

        // DatePickerDialog 생성(현재 날짜를 기본값으로 설정)
        DatePickerDialog dateDialog = new DatePickerDialog(this,
                                                new DatePickerDialog.OnDateSetListener() {
            // 사용자가 날짜를 선택했을 때 실행되는 콜백 메소드
```

```
            @Override
            public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
                  String date_msg = "선택한 날짜는" + year + "년"
                                    + (month + 1)  + "월" + dayOfMonth + "일";
                  tv_Date.setText(date_msg); // 선택한 날짜를TextView에 표시
            }
         }, mYear, mMonth, mDay);
         dateDialog.show(); // 다이얼로그 표시
      }

      // 시간 설정 다이얼로그를 띄우는 사용자 정의 메소드
      public void timeSet() {
         final Calendar cd = Calendar.getInstance(); // 현재 시간 가져오기
         int mHour = cd.get(Calendar.HOUR_OF_DAY);   // 시(24시간 기준)
         int mMinute = cd.get(Calendar.MINUTE);      // 분

         // TimePickerDialog 생성(현재 시간을 기본값으로 설정)
         TimePickerDialog timeDialog = new TimePickerDialog(this,
                                                   new TimePickerDialog.OnTimeSetListener() {
         // 사용자가 시간을 선택했을 때 실행되는 콜백 메소드
            @Override
            public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
               String time_msg = "선택한 시간은" + hourOfDay + "시" + minute + "분";
               v_Time.setText(time_msg); // 선택한 시간을TextView에 표시
            }
         }, mHour, mMinute, false); // false → 12시간제, true → 24시간제
         timeDialog.show(); // 다이얼로그 표시
      }
   }
```



날짜와 시간 다이얼로그는 프로젝트 생성시 지정된 테마 색상으로 표기가 된 것을 확인할 수 있을 것인데, 간단한 코드 추가만으로도 날짜와 시간 다이얼로그의 색상을 변경할 수 있다.

`new DatePickerDialog(this, new DatePickerDialog.OnDateSetListener() {` 부분을

`new DatePickerDialog(this, android.R.style.Theme_Material_Dialog, new DatePickerDialog.OnDateSetListener() {` 로 변경하여 어떻게 변화하는지 확인해 보세요.

마찬가지 방법으로 `new TimePickerDialog(this, new TimePickerDialog.OnTimeSetListener() {` 부분에도 동일하게 적용시켜 확인해 보시기 바랍니다.



연습문제

1. 안드로이드 앱에서 사용되는 옵션 메뉴, 컨텍스트 메뉴, 팝업 메뉴의 공통점과 차이점을 용도, 표시 위치, 활성화 방식을 중심으로 서술하고, 각 메뉴가 사용자 경험(UX)에 어떻게 기여하는지 구체적인 예를 들어 설명하세요.

2. onCreateOptionsMenu(), onOptionsItemSelected(), 그리고 onPrepareOptions Menu() 메서드의 각각의 역할과 호출 시점을 자세히 설명하세요. 또한, 동적 메뉴 구성 시 onPrepareOptionsMenu()가 왜 중요한지 그 이유를 코드 예시와 함께 서술하세요.

3. 안드로이드 앱의 상단 앱 바(App Bar)를 구성하는 ActionBar와 Toolbar의 주요 차이점을 도입 시기, 커스터마이징 가능성, 호환성 측면에서 비교하여 설명하세요. 최신 안드로이드 앱 개발에서 Toolbar를 사용하는 경향이 두드러지는 이유를 서술하세요.

4. 메뉴 리소스 XML 파일에서 〈group〉 태그를 사용하는 이유를 코드의 가독성 및 유지보수성 측면에서 설명하세요. 또한, android:checkableBehavior 속성이 single과 all일 때의 동작 방식 차이를 예시와 함께 서술하세요.

5. AlertDialog, DatePickerDialog, TimePickerDialog의 주요 용도와 사용 시점을 비교하여 설명하세요. 이 세 가지 대화상자 외에 사용자에게 완전히 커스터마이징된 레이아웃을 제공해야 할 때 어떤 방식으로 구현할 수 있는지 서술하세요.



제5장

애니메이션



학습목표

- View 애니메이션(트윈) · 프레임 애니메이션과 Property Animation(Object/ValueAnimator)의 차이를 구분 · 설명한다.
- ViewPropertyAnimator와 ObjectAnimator의 사용 시점 · 장단점을 비교한다.
- 애니메이션 동작 구현을 위한 XML을 작성할 수 있다.
- Java 코드로 복합 애니메이션 동작을 구현할 수 있다.

5.1 XML 활용 애니메이션

5.1.1 애니메이션 개요

애니메이션은 모바일 애플리케이션 개발에서 사용자 경험(UX)을 크게 향상시킬 수 있는 매우 강력한 도구입니다. 단순히 화면을 화려하게 만드는 데 그치지 않고, 사용자가 앱과 상호작용할 때 직관적인 시각적 피드백을 제공하며, 다양한 UI 요소의 변화를 부드럽게 연결해 주는 역할을 합니다. 이러한 과정을 통해 앱은 단순히 정적인 화면을 넘어, 보다 역동적이고 생동감 있는 인터페이스를 제공하게 되며, 결과적으로 사용자는 앱을 더욱 매력적이고 편리하게 느낄 수 있습니다.

애니메이션의 핵심은 '정적인 화면 요소에 시간의 흐름을 더한다'는 점에 있습니다. 즉, 버튼이나 이미지와 같은 고정된 UI 요소에 움직임을 부여하거나, 크기 · 위치 · 투명도 등 속성을 변화시킴으로써 동적인 효과를 만들어 내는 것입니다. 이렇게 구현된 애니메이션은 사용자의 시선을 자연스럽게 유도하고, 앱의 상태 변화를 시각적으로 이해하기 쉽게 전달하는 데에도 중요한 역할을 합니다.

안드로이드에서 제공하는 애니메이션 기법은 구현 방식과 특성에 따라 크게 세 가지 유형으로 구분됩니다.

- View Animation: 뷰의 크기, 위치, 회전, 투명도 등을 조절하여 애니메이션을 구현하는 방식으로, UI 요소의 단순한 시각적 변화를 연출하는 데 주로 활용됩니다.
- Drawable Animation: 여러 장의 이미지를 순차적으로 재생하여 마치 만화나 영상처럼 보이도록 만드는 방식으로, 간단한 캐릭터 동작이나 아이콘 효과 등에 적합합니다.
- Property Animation: 객체의 속성 값을 직접 변경하여 애니메이션 효과를 주는 방식으로, View Animation보다 훨씬 유연하고 다양한 효과를 구현할 수 있어 최근에는 가장 많이 활용되는 기법입니다.

표 5-1 애니메이션 방식 비교

애니메이션 방식	특징	활용 예
View Animation	위치/크기/회전/투명도 변경	화면 전환, 버튼 클릭 효과
Drawable Animation	프레임 이미지 생성	로딩 애니메이션, 캐릭터 동작
Property Animation	속성 값 직접 제어	UI 동적 반응, 복잡한 모션

5.1.2 View Animation (트윈 애니메이션)

View Animation은 안드로이드에서 가장 기본적으로 사용되는 애니메이션 방식 중 하나로, 주로 뷰(View)의 크기, 위치, 회전, 투명도와 같은 속성에 변화를 주어 시각적인 효과를 만들어 냅니다. 이 방식은 간단하면서도 직관적으로 애니메이션을 정의할 수 있다는 장점이 있으며, XML 파일을 활용해 손쉽게 설정할 수 있습니다.

View Animation을 사용할 때는 애니메이션을 정의한 XML 파일을 /res/anim 폴더에 저장해야 합니다. 이렇게 정의된 XML 파일은 레이아웃이나 코드에서 불러와 뷰 객체에 적용할 수 있으며, 다양한 화면 전환이나 강조 효과를 구현하는 데 널리 활용됩니다. 예를 들어 버튼을 눌렀을 때 살짝 커졌다가 작아지는 효과, 이미지가 천천히 사라지는 페이드아웃 효과 등을 View Animation으로 간단히 만들 수 있습니다.

이때 XML 파일 내에서는 특정한 태그를 사용하여 애니메이션의 종류와 속성을 지정하게 되는데, 대표적으로는 〈translate〉(이동), 〈scale〉(크기 변화), 〈rotate〉(회전), 〈alpha〉(투명도 조절)와 같은 태그들이 있습니다. 각 태그에는 시작 값과 끝 값, 애니메이션이 지속되는 시간, 반복 여부 등의 속성을 설정할 수 있으며, 여러 태그를 조합하여 보다 복잡한 애니메이션 효과를 구현할 수도 있습니다.

따라서 View Animation은 앱의 화면 전환이나 버튼 · 이미지와 같은 UI 요소에 생동감을 더해 사용자 경험을 향상시키는 데 매우 효과적인 기법이며, XML 기반으로 구조화되어 있기 때문에 코드와 레이아웃을 분리해 관리할 수 있다는 장점도 가지고 있습니다.
View Animation의 주요 속성 태그는 아래와 같습니다.

- 〈translate〉: 위치 이동
- 〈scale〉: 크기 변경
- 〈rotate〉: 회전
- 〈alpha〉: 투명도 변경
- 〈set〉: 여러 애니메이션을 묶어 실행

다음은 View Animation을 위한 XML 파일 예시입니다.

```
// move_rotate.xml

<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_decelerate_interpolator"
    android:duration="2000">
    <translate
        android:fromXDelta="0%"
        android:toXDelta="50%"
        android:fromYDelta="0%"
        android:toYDelta="50%" />
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%" />
</set>
```

위와 같이 정의된 애니메이션 XML 파일을 아래와 같이 애니메이션 동작시킬 객체에 적용시키면 됩니다.

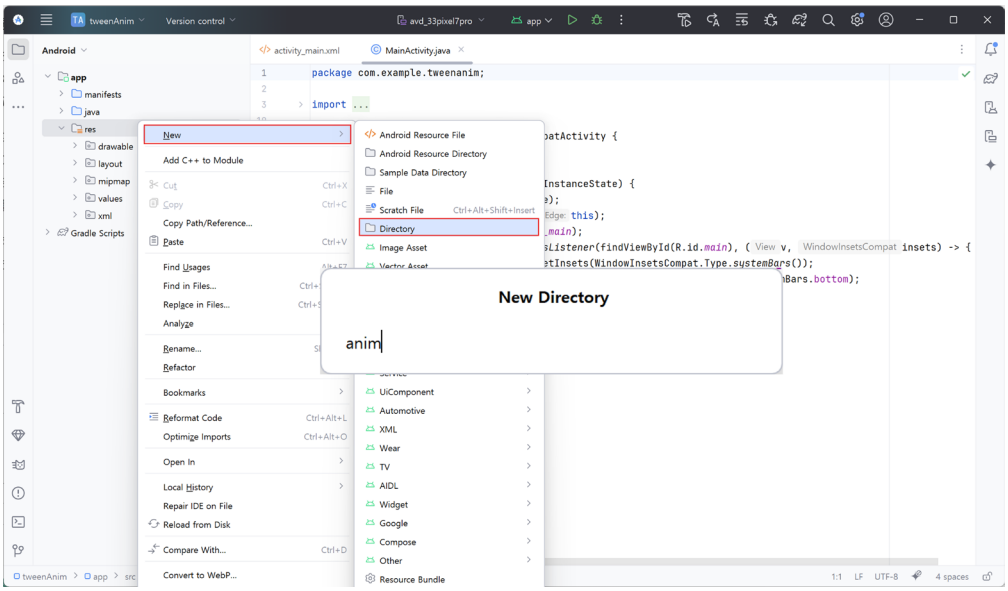
```
ImageView imageView = findViewById(R.id.imageView);
Animation anim = AnimationUtils.loadAnimation(this, R.anim.move_rotate);
imageView.startAnimation(anim);
```

다음은 View Animation (트윈 애니메이션) 실습 예제입니다. 다음과 같이 설정하여 프로젝트를 생성하세요.

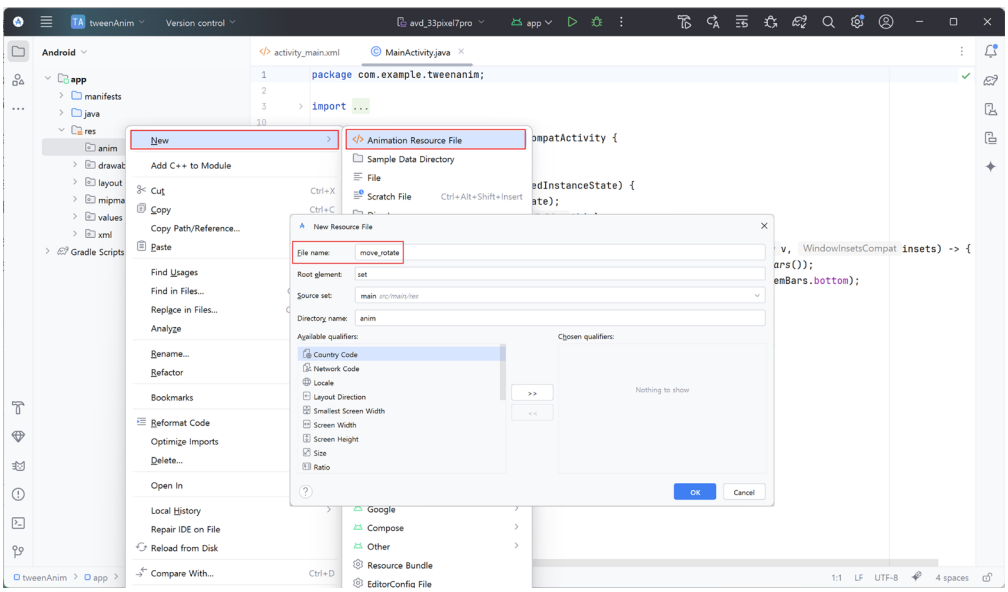
- Project Name : tweenAnim
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) XML 생성 및 작성

Project Navigator(View) 창에서 app 〉 res 폴더를 선택하고 마우스 우클릭 한 후, New 〉 Directory를 클릭하고 "anim" 입력하여 폴더를 생성하세요.



다음으로, 생성된 anim 폴더를 우클릭 후, new 〉 Animation Resource File을 선택한 후 "move_rotate" 입력하여 파일을 생성하세요.



파일이 생성된 후, 아래와 같이 입력하세요.

move_rotate.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="2000"
    android:interpolator="@android:anim/accelerate_decelerate_interpolator"
    android:fillAfter="true">

    <!-- 위치 이동 -->
    <translate
        android:fromXDelta="0%"
        android:toXDelta="50%"
        android:fromYDelta="0%"
        android:toYDelta="50%" />

    <!-- 회전 -->
    <rotate
        android:fromDegrees="0"
        android:toDegrees="360"
        android:pivotX="50%"
        android:pivotY="50%" />

</set>
```

XML에 기재된 주요 속성은 아래와 같습니다.

- android:duration="2000"〉: 애니메이션 동작 2초 동안 유지
- android:fillAfter="true" : 애니메이션 종료 후 상태 유지
- 〈translate〉: X, Y 좌표를 % 기준으로 이동
- 〈roate〉: 뷰의 중심점을 기준으로 회전. pivotX, pivotY를 통해 중심점 변경 가능

XML 기재된 애니메이션 동작은

- 〈translate〉 태그:
  - 위치이동을 의미하며, 현 위치에서 X축 방향(fromXDelta, toXDelta)으로 지정된 % 만큼 이동 (단, 뷰 크기의 %만큼)
  - Y축 방향(fromYDelta, toYDelta)으로 지정된 % 만큼 이동

- 〈rotate〉 태그:
  - 0도에서 시계방향으로 한 바퀴(360) 회전
  - pivotX=50%, pivotY=50%: 회전 중심점을 뷰의 가로, 세로 중앙으로 지정

으로 설정되어 있습니다.

즉, 뷰가 2초 동안 오른쪽 아래 방향으로 이동하면서 동시에 360도 회전이 끝난 후에는 이동한 위치와 회전상태를 그대로 유지합니다.

(2) Layout(레이아웃) XML 작성

레이아웃 XML 파일을 [CODE] 모드로 변경한 후 아래와 같이 작성하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/face02"
        android:layout_centerInParent="true" />

    <Button
        android:id="@+id/btnStart"
        android:layout_width="wrap_content"
        android:layout_height=""wrap_content"
        android:text="애니메이션 시작"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="30dp" />

</RelativeLayout>
```

추가로 다운로드 받은 이미지 중 “face02.png” 파일을 app 〉 res 〉 drawable 폴더에 복사 및 붙여넣기 해 주세요.

(3) 코드 작성

View Animation(트윈) 애니메이션 동작을 위한 애니메이션 XML과 화면 디자인 XML 파일 작업이 완료되었으면 아래와 같이 코드를 작성하세요.

MainActivity.java

```
package com.example.tweenanim;

import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

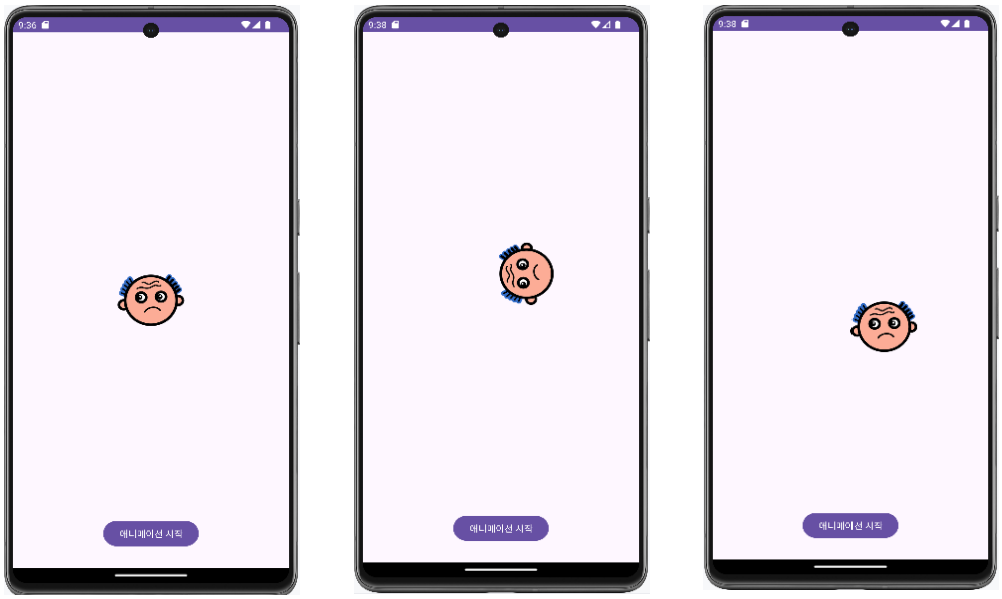
public class MainActivity extends AppCompatActivity {

    ImageView imageView;
    Button btnStart;

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

        imageView = findViewById(R.id.imageView);
        btnStart = findViewById(R.id.btnStart);

        btnStart.setOnClickListener(v -> {
        Animation anim = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.move_rotate);
        imageView.startAnimation(anim);
        });
    }
}
```



5.1.3 Drawable Animation (프레임 애니메이션)

드로어블 애니메이션(Drawable Animation)은 흔히 프레임 애니메이션(Frame Anima- tion)이라고도 불리며, 여러 장의 이미지를 순차적으로 빠르게 전환하여 마치 하나의 움직이는 장면처럼 보이도록 구현하는 애니메이션 기법입니다. 이 방식은 애니메이션의 각 장면을 이미지로 미리 준비해 놓고, 이를 일정한 시간 간격으로 연속적으로 보여줌으로써 동적인 효과를 만들어내는 원리입니다.

예를 들어 어린 시절 책의 모서리마다 조금씩 다른 그림을 그리고 책장을 빠르게 넘기면, 그림이 살아 움직이는 듯한 효과를 볼 수 있었던 경험이 있을 것입니다. 드로어블 애니메이션은 바로 이러한 원리를 디지털 환경에서 재현한 것이라고 할 수 있습니다.

안드로이드에서는 이러한 프레임 기반의 애니메이션을 AnimationDrawable 클래스를 통해 손쉽게 구현할 수 있습니다. 개발자는 여러 장의 이미지 리소스를 준비하고, XML 파일을 통해 각 이미지(프레임)가 표시될 순서와 지속 시간을 정의합니다. 이후 코드에서 해당 XML을 불러와 실행하면, 지정된 시간 간격에 따라 이미지가 자동으로 전환되면서 자연스러운 애니메이션 효과가 완성됩니다.

이 기법은 비교적 단순한 구조로 인해 캐릭터의 움직임, 로딩 화면, 간단한 아이콘 효과 등 직

관적이고 시각적으로 눈에 띄는 애니메이션을 구현하는 데 널리 활용됩니다. 다만 많은 이미지를 사용하면 앱 용량이 커지거나 메모리 사용량이 증가할 수 있으므로, 주로 짧고 반복적인 애니메이션에서 효과적으로 사용되는 방식입니다.

다음은 Drawable Animation 예시 파일입니다.

```
// frame_anim.xml

<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/frame1" android:duration="200"/>
    <item android:drawable="@drawable/frame2" android:duration="200"/>
    <item android:drawable="@drawable/frame3" android:duration="200"/>
</animation-list>
```

```
// MainActivity.java

ImageView imageView = findViewById(R.id.imageView);
imageView.setBackgroundResource(R.drawable.frame_anim);
AnimationDrawable animDrawable = (AnimationDrawable) imageView.getBackground();
animDrawable.start();
```

위 예시 xml 파일처럼 @drawable 폴더 내에 여러 이미지를 일정 유지시간동안 보여주고 다음 이미지로 넘기는 방식으로 지정되어 있고, 액티비티 내 코드를 통해 동작한다. animation-list 내 android:oneshot 속성 값이 "true" 인 경우에는 애니메이션 동작을 전체 순회 1회 후 종료되게 되며, "false"인 경우 계속 반복하게 된다.

다음은 Drawable Animation (프레임 애니메이션) 실습 예제입니다. 다음과 같이 설정하여 프로젝트를 생성하세요.

- Project Name : frameAnim
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

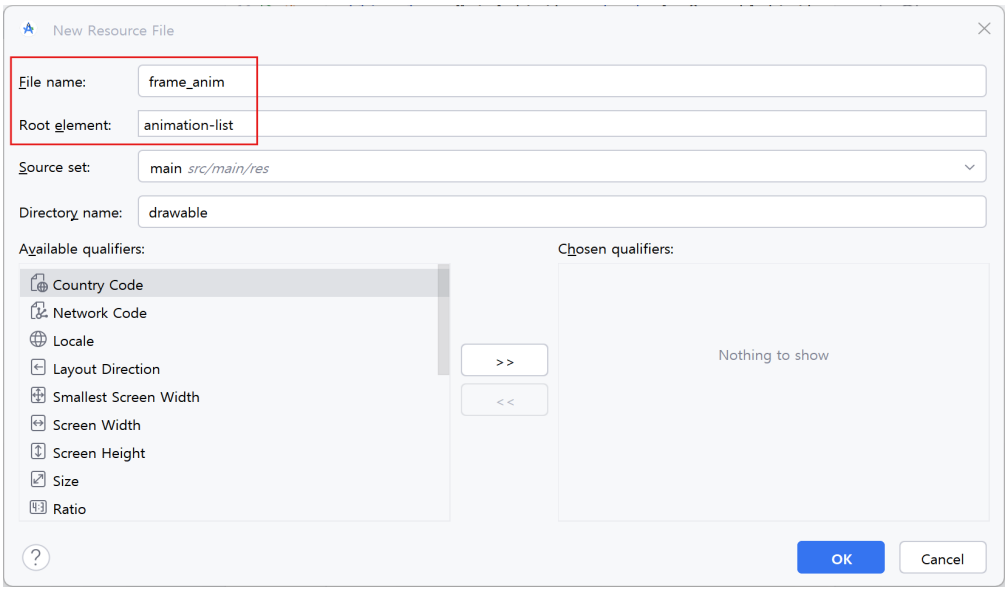
(1) XML 생성 및 작성

앞에서 살펴보았던 View Animation과 다르게 app 〉 res 〉 anim 폴더 내에 애니메이션 정의 XML을 생성시키는 것이 아니라 drawable 폴더 내에 애니메이션 정의 XML을 생성시킵니다.

먼저, ① 다운로드 받은 이미지 중 애니메이션 정의에 필요한 이미지 (walk01~walk10)를 app 〉 res 〉 drawable 에 복사/붙여넣기(Ctrl + c, v) 하세요.

다음으로 ② app 〉 res 〉 drawable 폴더에서 마우스 우클릭 후, new 〉 Drawable Resource File을 선택하세요.

"File Name"은 frame_anim, "Root element" 는 "animation-list" 로 지정하세요.



XML 파일 생성후, 기본 [Design] 모드를 [Code] 로 변경하여 아래 코드를 입력하세요.

frame_anim.xml

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">

    <!-- 각 프레임 정의(이미지와 지속 시간) -->
    <item android:drawable="@drawable/walk01" android:duration="200" />
    <item android:drawable="@drawable/walk02" android:duration="200" />
    <item android:drawable="@drawable/walk03" android:duration="200" />
    <item android:drawable="@drawable/walk04" android:duration="200" />
    <item android:drawable="@drawable/walk05" android:duration="200" />
```

```
    <item android:drawable="@drawable/walk06" android:duration="200" />
    <item android:drawable="@drawable/walk07" android:duration="200" />
    <item android:drawable="@drawable/walk08" android:duration="200" />
    <item android:drawable="@drawable/walk09" android:duration="200" />
    <item android:drawable="@drawable/walk10" android:duration="200" />
</animation-list>
```

(2) Layout(레이아웃) XML 작성

레이아웃 XML 파일을 [CODE] 모드로 변경한 후 아래와 같이 작성하세요.

MainActivity.java

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="30dp"
        android:text="클릭" />

    <ImageView
        android:id="@+id/iv_main"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_centerInParent="true"
        android:background="@drawable/frame_anim" />

</RelativeLayout>
```

(3) 코드 작성

Drawable Animation 애니메이션 동작을 위한 애니메이션 XML과 화면 디자인 XML 파일 작업이 완료되었으면 아래와 같이 코드를 작성하세요.

MainActivity.java

```
package com.example.frameanim;

import android.graphics.drawable.AnimationDrawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

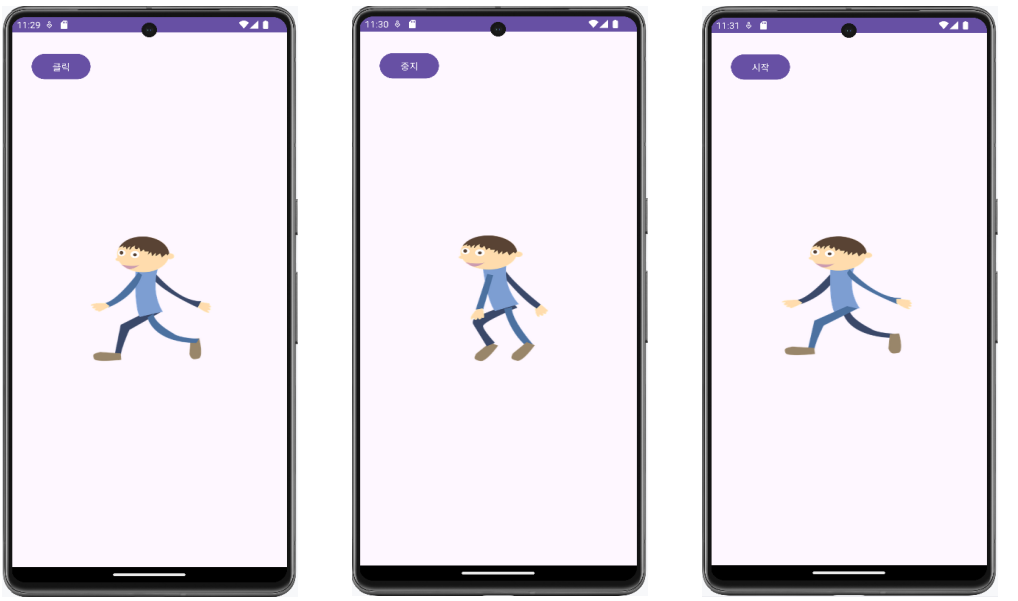
        final Button btn_s = (Button) findViewById(R.id.btn_start);
        ImageView iv_m = (ImageView) findViewById(R.id.iv_main);

        final AnimationDrawable walking = (AnimationDrawable)
                                                iv_m.getBackground();

        btn_s.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(walking.isRunning()) {
                    walking.stop();
                    btn_s.setText("시작");
                } else {
                    walking.start();
                    btn_s.setText("중지");
                }
            }
        });
    }
}
```

코드 설명

- final AnimationDrawable walking = (AnimationDrawable) iv_m.getBackground();
  layout(레이아웃) XML에서 ImageView 객체 속성 중
  android:background="@drawable/frame_anim" 에 지정된 애니메이션 XML 정의를
  AnimationDrawable 객체변수인 walking 에 로드한다.
- btn_s.setOnclickListener()를 통해 버튼 클릭시 아래 동작을 수행한다.
- if(walking.isRunning()) {: 애니메이션 동작 상태일 때
  - walking.stop(): 애니메이션 재생멈춤
  - btn_s.setText("시작"): 버튼 객체의 텍스트를 "시작"으로 변경
- else {: 애니메이션 정지 상태일 때
  - walking.start(): 애니메이션 재생시작
  - btn_s.setText("중지"): 버튼 객체의 텍스트를 "중지"로 변경



5.1.4 Interpolator

Interpolator(인터폴레이터)는 안드로이드 애니메이션에서 시간에 따른 속도 변화의 곡선을 정의하는 객체로, 애니메이션이 시작부터 끝까지 동일한 속도로 진행되는 것이 아니라 속도의 가감

속을 조절할 수 있도록 도와주는 중요한 요소입니다. 쉽게 말해, 애니메이션의 "리듬"과 "타이밍"을 결정하는 역할을 한다고 볼 수 있습니다.

예를 들어, 어떤 애니메이션을 처음에는 천천히 시작했다가 점점 빨라지게 하거나, 반대로 빠르게 시작했다가 서서히 느려지도록 만들고 싶을 때 Interpolator를 활용합니다. 또한 특정 시점에서 튕기듯이 움직이거나, 탄성 효과를 주는 등 다양한 속도 변화를 적용할 수도 있습니다.

안드로이드에서 제공하는 대표적인 Interpolator에는 LinearInterpolator(항상 일정한 속도로 진행), AccelerateInterpolator(처음에는 느리게 시작하다가 점점 빨라짐), DecelerateInterpolator(빠르게 시작하여 점점 느려짐), AccelerateDecelerateInterpolator(처음과 끝은 느리고 중간은 빠름) 등이 있습니다. 또한 개발자가 직접 커스텀 Interpolator를 구현하여 독창적인 속도 변화를 적용할 수도 있습니다.

즉, Interpolator는 애니메이션의 결과 자체를 바꾸는 것이 아니라, 시간의 흐름에 따라 애니메이션이 재생되는 속도와 느낌을 결정하여 사용자에게 더욱 자연스럽고 매력적인 시각적 경험을 제공하는 도구라고 할 수 있습니다.

(1) Interpolator의 역할

Interpolator는 안드로이드 애니메이션에서 단순히 애니메이션이 일정 시간 동안 실행되도록 하는 것에 그치지 않고, 애니메이션의 재생 속도가 시간에 따라 어떻게 변화할지를 정의하는 핵심적인 역할을 담당합니다. 보통 애니메이션은 duration 속성을 통해 전체 실행 시간을 지정할 수 있지만, 이 속성만으로는 속도의 변화나 리듬감을 표현할 수 없습니다.

예를 들어, 2초 동안 이동하는 애니메이션이 있다고 할 때, 단순히 duration=2000ms만 설정하면 처음부터 끝까지 똑같은 속도로만 실행됩니다. 그러나 실제 사용자 경험을 고려한다면, 시작할 때는 천천히 움직이다가 중간에는 빠르게 진행하고, 마지막에는 서서히 멈추는 것이 훨씬 더 자연스럽고 직관적인 효과를 줍니다.

이러한 속도 변화를 가능하게 해주는 것이 바로 Interpolator 입니다. Interpolator 를 사용하면 애니메이션의 시작 부분, 중간 부분, 끝 부분의 속도를 각각 다르게 설정할 수 있어, 애니메이션이 단순한 이동이나 변화가 아니라 마치 실제 물체가 움직이는 듯한 자연스러움을 구현할 수 있습니다. 즉, Interpolator는 애니메이션의 시간 축을 조절하여, 일정한 시간 동안 진행되는 동작이 어떻게 가속되거나 감속되는지를 정의함으로써 앱의 사용자 경험을 한층 더 생동감 있게 만들어 줍니다.

Interpolator의 역할을 정리하면 다음과 같습니다.

- 애니메이션의 재생속도 패턴을 정의
- duration(유지기간) 정의로 속도변화 표현
- 애니메이션의 시작, 중간, 끝 부분 속도를 다르게 조절 가능

(2) 주요 Interpolator 종류 및 특징

안드로이드에서 제공하는 Interpolator는 애니메이션의 속도 변화를 제어하는 핵심 도구로, 단순히 일정한 속도로 움직이는 효과를 넘어 자연스러운 동작과 사용자 경험의 향상을 가능하게 합니다. 예를 들어, 물체가 처음에는 천천히 움직이다가 점점 빨라지거나, 빠르게 출발했다가 끝으로 갈수록 서서히 멈추는 동작을 구현할 수 있습니다. 이러한 효과는 실제 물리적 움직임과 유사한 경험을 제공하여 사용자에게 친숙하고 직관적인 느낌을 줍니다.

안드로이드에는 다양한 종류의 Interpolator가 존재하며, 각각 고유한 속도 곡선을 정의합니다. 예를 들어, LinearInterpolator는 일정한 속도로 움직이는 단순한 패턴을 제공하고, AccelerateInterpolator는 점점 빨라지는 효과를, DecelerateInterpolator는 반대로 점점 느려지는 효과를 구현합니다. 또한 AccelerateDecelerateInterpolator는 처음과 끝은 느리게, 중간은 빠르게 움직이도록 하여 부드럽고 자연스러운 애니메이션을 연출할 수 있습니다. 이 외에도 BounceInterpolator나 OvershootInterpolator처럼 튕기는 듯한 효과나 목표 지점을 살짝 넘어갔다가 돌아오는 효과를 줄 수 있는 특수한 형태도 있습니다.

따라서 애니메이션을 설계할 때 단순히 지속 시간(duration)만 설정하는 것이 아니라, Interpolator의 종류와 특징을 충분히 이해한 뒤 상황에 맞게 선택하여 적용하는 것이 필요합니다. 이렇게 하면 앱의 UI가 보다 현실감 있고 생동감 있는 동작을 보여줄 수 있으며, 사용자는 앱의 기능뿐만 아니라 시각적 · 심리적으로도 긍정적인 경험을 하게 됩니다.

표 5-2 주요 Interpolator 종류와 특징

Interpolator	설명
LinearInterpolator	일정 속도로 진행
AccelerateInterpolator	느리게 시작 → 점점 빨라짐
DecelerateInterpolator	빠르게 시작 → 점점 느려짐
AccelerateDecelerateInterpolator	느리게 시작 → 빨라짐 → 느려짐
BounceInterpolator	끝에서 튕기는 효과
OvershootInterpolator	목표 지점을 살짝 넘어갔다가 돌아옴
AnticipateInterpolator	시작 전에 반대 방향으로 살짝 당겼다가 진행
AnticipateOvershootInterpolator	시작 전 당기고, 목표 넘어서고 돌아옴

(3) 활용 방법

Interpolator를 애니메이션에 적용하는 방법에는 크게 두 가지가 있습니다. 첫 번째는 XML 리소스 파일에서 Interpolator를 정의하고 적용하는 방식이고, 두 번째는 자바(Java)나 코틀린(Kotlin) 코드에서 직접 Interpolator 객체를 생성하여 애니메이션에 적용하는 방식입니다.

XML을 활용하는 방식은 프로젝트의 /res/anim 폴더 내에 정의된 애니메이션 리소스 파일에 android:interpolator 속성을 지정하는 방법입니다. 이 방식은 코드와 디자인을 분리할 수 있어 유지보수성과 재사용성이 높다는 장점이 있습니다.

예를 들어, res/anim/move.xml과 같은 애니메이션 리소스에 android:interpolator="@android:anim/accelerate_interpolator"를 지정하면, 해당 애니메이션이 실행될 때 자동으로 가속 효과가 적용됩니다.

반면, 코드에서 직접 적용하는 방식은 애니메이션 객체를 생성한 뒤 set Interpolator() 메서드를 호출하여 원하는 Interpolator를 지정하는 방법입니다. 예를 들어, TranslateAnimation이나 ObjectAnimator 객체에 animation.setInterpolator(new Bounce Interpolator());와 같이 적용하면, 그 애니메이션이 실행될 때 튕기는 효과가 나타나게 됩니다. 이 방식은 상황에 따라 동적으로 Interpolator를 바꿀 수 있다는 장점이 있으며, 사용자 동작이나 특정 조건에 따라 유연하게 애니메이션을 제어할 수 있습니다.

즉, Interpolator 적용은 XML을 통한 정적 정의 방식과 코드 기반의 동적 제어 방식으로 나눌 수 있으며, 프로젝트의 성격이나 구현 목적에 따라 적절한 방법을 선택하여 활용하면 애니메

이션 효과를 보다 다양하고 직관적으로 연출할 수 있습니다.

각 방식에 대해 적용 예시방식을 확인해 보세요.

❶ XML에서 적용

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/bounce_interpolator">
    <translate
        android:fromYDelta="0%"
        android:toYDelta="50%"
        android:duration="1000" />
</set>
```

❷ 코드에서 적용

```
Animation anim = AnimationUtils.loadAnimation(this, R.anim.move);
anim.setInterpolator(new BounceInterpolator());
view.startAnimation(anim);
```

(4) Interpolator 실습

Interpolator 실습을 위해 아래와 같이 프로젝트를 생성하세요.

- Project Name : interpolatorExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

프로젝트 생성 후, 애니메이션 XML정의, 레이아웃 정의, 코드를 아래와 같이 각각 작성하세요.

❶ 애니메이션 XML 정의 (app 〉 res 〉 anim 〉 move.xml)

move.xml

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="2000"
    android:fromXDelta="0%"
    android:toXDelta="70%"
    android:fromYDelta="0%"
    android:toYDelta="0%" />
```

❷ 레이아웃 XML (app 〉 res 〉 layout 〉 activity_main.xml)

activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:orientation="vertical">

  <Button
    android:id="@+id/btnLinear"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Linear Interpolator"
    android:textSize="25dp" />

  <Button
    android:id="@+id/btnAccelerate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Accelerate Interpolator"
    android:textSize="25dp" />

  <Button
    android:id="@+id/btnDecelerate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Decelerate Interpolator"
    android:textSize="25dp" />

  <Button
    android:id="@+id/btnBounce"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Bounce Interpolator"
    android:textSize="25dp" />

</LinearLayout>
```

❸ 코드 작성 (MainActivity.java)

MainActivity.java

```
package com.example.interpolatorexam;

import android.os.Bundle;
import android.view.animation.AccelerateInterpolator;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.view.animation.BounceInterpolator;
import android.view.animation.DecelerateInterpolator;
import android.view.animation.LinearInterpolator;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    // 4가지 종류의Interpolator 효과를 확인하기 위한 버튼 선언
    Button btnLinear, btnAccelerate, btnDecelerate, btnBounce;

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

        // XML에 정의된 버튼 객체 연결
        btnLinear = findViewById(R.id.btnLinear);
        btnAccelerate = findViewById(R.id.btnAccelerate);
        btnDecelerate = findViewById(R.id.btnDecelerate);
        btnBounce = findViewById(R.id.btnBounce);

        // 각 버튼 클릭 시, 해당Interpolator를 적용한 애니메이션 실행
        // 선형적으로 속도가 일정하게 유지되는Interpolator
        btnLinear.setOnClickListener(v -> startAnim(btnLinear, new LinearInterpolator()));

        // 시작은 천천히, 끝으로 갈수록 점점 빨라지는 가속형Interpolator
        btnAccelerate.setOnClickListener(v -> startAnim(btnAccelerate, new
                                                    AccelerateInterpolator()));
        // 시작은 빠르지만, 끝으로 갈수록 점점 느려지는 감속형Interpolator
        btnDecelerate.setOnClickListener(v -> startAnim(btnDecelerate, new
                                                    DecelerateInterpolator()));
        // 바운스(튕김) 효과가 적용되는Interpolator
        btnBounce.setOnClickListener(v -> startAnim(btnBounce, new BounceInterpolator()));
}
```

```
    /**
     * 공통 애니메이션 실행 메서드
     * @param button : 애니메이션을 적용할 버튼
     * @param interpolator : 적용할 Interpolator 객체
     */
    private void startAnim(Button button, android.view.animation.Interpolator interpolator) {
        // /res/anim/move.xml에 정의된 애니메이션 불러오기
        Animation anim = AnimationUtils.loadAnimation(this, R.anim.move);

        // 전달받은Interpolator를 애니메이션에 적용
        anim.setInterpolator(interpolator);

        // 버튼에 애니메이션 실행
        button.startAnimation(anim);
    }
}
```

5.2 코딩 활용 애니메이션

XML을 활용한 View 애니메이션(Tween Animation)이나 Drawable 애니메이션(Frame Animation)은 주로 선언적 방식으로 정의되며, 단순하고 정적인 효과를 구현하는 데 적합합니다. 예를 들어, 화면 전환 시 부드럽게 페이드 인 · 아웃 효과를 주거나, 버튼 클릭 시 살짝 커졌다 작아지는 단순한 모션을 구현하는 데 효과적입니다. 그러나 이러한 XML 기반 애니메이션은 기본적으로 정해진 패턴대로만 동작하기 때문에, 동적으로 변하는 조건이나 복잡한 상태 변화에는 한계가 있습니다.

이에 비해 코딩을 기반으로 구현하는 애니메이션 기법은 상황에 따라 훨씬 더 유연하고 강력한 기능을 제공합니다. 사용자의 입력 동작, 센서 값의 변화, 네트워크로부터 전달되는 데이터와 같은 외부 요인에 따라 실시간으로 애니메이션 동작을 제어할 수 있으며, 단순히 View 요소에만 한정되지 않고 임의의 객체 속성에도 애니메이션 효과를 적용할 수 있습니다.

또한 코딩 기반 애니메이션은 시간 제어, 가속/감속 곡선(Interpolator) 적용, 반복 실행이나 역재생, 여러 속성을 동시에 변화시키는 복합 모션, 키프레임을 활용한 세밀한 모션 제어와 같은 고급 기능들을 지원합니다. 이로 인해 단순한 시각적 효과를 넘어, 앱 전반의 사용자 경험을 세밀하게 디자인할 수 있습니다.

안드로이드에서는 이러한 코딩 기반 애니메이션을 구현하기 위해 Property Animation과

ObjectAnimator를 제공합니다. Property Animation은 객체의 특정 속성을 지정한 시간 동안 원하는 값으로 변화시키는 방식을 사용하며, ObjectAnimator는 이를 손쉽게 다룰 수 있도록 API 형태로 제공되는 대표적인 클래스입니다. 따라서 정적이고 단순한 효과는 XML 기반으로, 복잡하고 동적인 효과는 코드 기반의 Property Animation 기법으로 구현하는 것이 일반적입니다.

5.2.1 Property Animation

Property Animation은 안드로이드에서 제공하는 강력한 애니메이션 기법으로, 단순히 화면에 보여지는 View 요소의 모양이나 위치를 바꾸는 수준을 넘어, 객체가 가지고 있는 속성(Property)의 값을 직접 변경하여 애니메이션 효과를 구현합니다. 이 방식은 특정 속성이 시간의 흐름에 따라 원하는 값으로 점진적으로 변화하도록 제어하기 때문에, 단순한 이동 · 회전 · 확대 축소 같은 효과뿐만 아니라, 색상, 투명도, 수치 데이터 등 다양한 속성에 적용할 수 있습니다.

또한 Property Animation은 View 객체에만 한정되지 않고, 모든 종류의 객체에 적용할 수 있다는 점에서 매우 유연합니다. 예를 들어, 화면에 배치된 버튼이나 텍스트뷰의 위치와 크기를 변화시키는 것은 물론, 사용자 정의 클래스의 속성 값이나 비주얼 요소와 직접적으로 연결되지 않은 데이터 값까지도 애니메이션의 대상으로 만들 수 있습니다. 이 덕분에 Property Animation은 애니메이션을 통해 단순히 시각적 효과를 주는 수준을 넘어서, 앱의 전반적인 동작 로직과 유기적으로 연결된 동적이고 복합적인 사용자 경험을 구현하는 데 널리 활용됩니다.

Property Animation 실습을 위해 아래와 같이 프로젝트를 생성하세요.

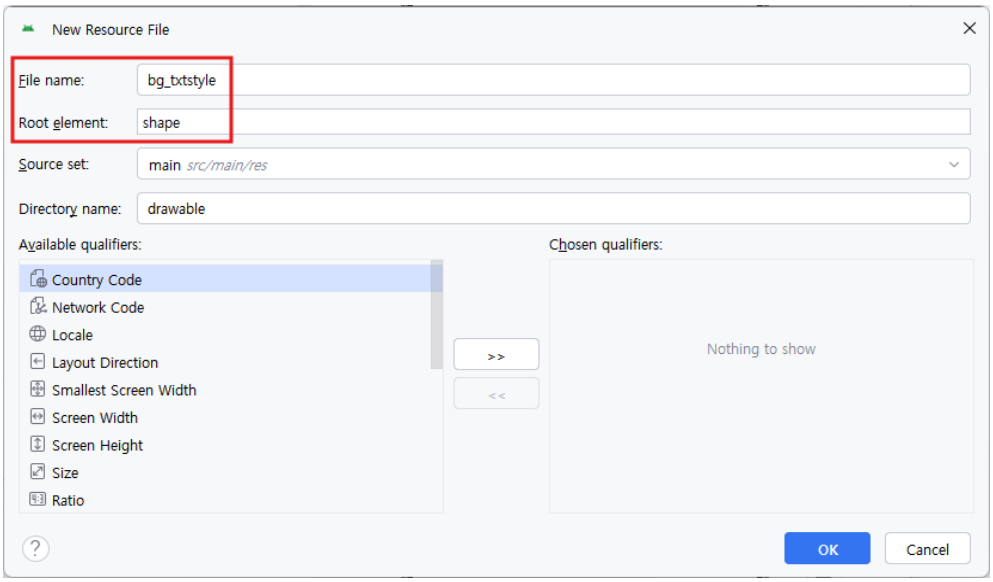
- Project Name : propertyExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) Style XML과 레이아웃 XML

안드로이드에서는 Layout 디자인에서 뷰/위젯(component)의 스타일(모양, 색상)을 직접 변경도 가능하지만, Style XML을 이용해 디자인 할 수도 있습니다.

app 〉 res 〉 drawable 폴더를 우클릭 한 후, new 〉 Drawable Resource File을 선택하고, 아래와 같이 설정하여 파일을 생성하세요.

- File Name : bg_txtStyle
- Root element : shape



파일 생성 후, [Code] 모드로 변경한 후, 아래와 같이 작성하세요.

bg_txtstyle.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <solid
        android:color="#4F89F3" />   <!- 색상 지정 ->

    <padding    .
        android:left="45px"
        android:top="15px"
        android:right="45px"
        android:bottom="15px" />  <!- 여백 지정 ->

    <corners
        android:topRightRadius="30px"
        android:bottomRightRadius="30px" />  <!- 우측상단과 하단 곡률 지정 ->

</shape>
```

다음으로 Layout을 [Code] 모드로 변경한 후 아래와 같이 작성하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="150dp"
        android:layout_marginTop="50dp"
        android:text="애니메이션 시작"
        android:textSize="25dp" />

    <TextView
        android:id="@+id/tv_Title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="30dp"
        android:layout_marginTop="300dp"
        android:background="@drawable/bg_txtstyle"  <!- style 지정 ->
        android:text="인공지능"
        android:textColor="@color/white"
        android:textSize="25dp" />

</RelativeLayout>
```

(2) 코드 작성

MainActivity.java

```
package com.example.propertyexam;

import android.os.Bundle;
import android.view.View;
import android.view.animation.BounceInterpolator;
import android.view.animation.LinearInterpolator;
import android.widget.Button;
import android.widget.TextView;
```

```
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    TextView title;

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

        Button btn_ai = (Button) findViewById(R.id.btn_start);
        title = (TextView) findViewById(R.id.tv_Title);

        btn_ai.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                title.animate()
                    .translationX(500f)  // X 축으로 이동
                    .setDuration(2500)   // 2500ms 동안
                    .setInterpolator(new BounceInterpolator())

                   // 위 애니메이션 실행 후 마지막으로 아래 애니메이션 실행
                   // 점진적으로 흐려지게
                    .withEndAction(new Runnable() {
                        @Override
                        public void run() {
                            title.animate()
                                .alpha(0f).setDuration(1500)
                                .setInterpolator(new LinearInterpolator())
                                .start();
                        }
                    })
                    .start();
            }
        });

    }
}
```

입력 완료 후, 실행하여 실행결과를 확인하세요.

5.2.2 ObjectAnimator Animation

ObjectAnimator는 안드로이드에서 제공하는 Property Animation(속성 애니메이션) 시스템의 핵심 클래스 중 하나로, UI 요소의 크기, 위치, 회전, 투명도와 같은 실제 속성 값을 일정한 시간 동안 점진적으로 변화시켜주는 역할을 합니다. 기존의 뷰 애니메이션(View Animation) 방식은 단순히 화면에 보여지는 모양만 변경하여 애니메이션처럼 보이게 만들었을 뿐, 뷰가 가지고 있는 실제 속성 값 자체는 변하지 않았기 때문에 애니메이션이 종료되면 원래 상태로 돌아오거나, 시각적인 효과와 실제 동작이 일치하지 않는 경우가 많았습니다.

반면, ObjectAnimator는 뷰의 속성을 직접 변경하기 때문에 애니메이션이 완료된 후에도 변경된 속성이 그대로 유지된다는 점에서 큰 차이를 보입니다. 예를 들어, 버튼을 오른쪽으로 이동시키는 애니메이션을 ObjectAnimator로 구현하면, 화면에서 단순히 이동된 것처럼 보이는 것이 아니라 실제로 버튼의 위치 속성(translationX 등)이 변경되며, 애니메이션 종료 후에도 버튼은 새로운 위치에 머물러 있습니다. 따라서 버튼을 클릭할 경우에도 그 새로운 위치에서 정상적으로 동작하게 되어, UI의 일관성과 자연스러운 사용자 경험을 제공합니다.

이러한 특징 덕분에 ObjectAnimator는 단순한 시각적 효과를 넘어, 애플리케이션의 UI를 동적으로 제어하고, 앱의 흐름에 맞는 자연스럽고 지속적인 사용자 인터페이스를 구축하는 데 널리 활용됩니다. 특히 Property Animation 시스템은 다양한 보간(Interpolator), 반복 및 역재생 설정, 여러 애니메이션의 조합 등도 지원하므로, 개발자는 보다 정교하고 매력적인 애니메이션을 구현할 수 있습니다.

따라서, 이번에는 ObjectAnimator 애니메이션 실습을 위해 새로운 프로젝트를 생성하세요.

- Project Name : objectAnimatorExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

다음으로, Layout XML과 뷰/객체 Style XML은 Property Animation에서 사용한 파일을 참조하여 작성하고, 코드만 변경하도록 하겠습니다.

MainActivity.java

```
package com.example.objectanimatorexam;

import android.animation.Animator;
```

```
import android.animation.AnimatorListenerAdapter;
import android.animation.ObjectAnimator;
import android.os.Bundle;
import android.view.View;
import android.view.animation.BounceInterpolator;
import android.view.animation.CycleInterpolator;
import android.view.animation.LinearInterpolator;
import android.widget.Button;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    // 애니메이션 객체 선언(이동, 회전, 확대/축소, 투명도 효과)
    ObjectAnimator tranx, trany, rx, ry, scalex, scaley, alpha;

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

        // 버튼과 텍스트뷰 연결
        Button btn_start = (Button) findViewById(R.id.btn_start);
        TextView title = (TextView) findViewById(R.id.tv_Title);

        /** --------이동 애니메이션 --------- */
        // X축 방향 이동(0 → 500px)
        tranx = ObjectAnimator.ofFloat(title, "translationX", 0F, 500F);
        tranx.setInterpolator(new BounceInterpolator()); // 튕기는 효과
        tranx.setDuration(2500); // 2.5초 동안 실행
        tranx.setRepeatCount(0); // 반복 없음

        // Y축 방향 이동(0 → 300px)
        trany = ObjectAnimator.ofFloat(title, "translationY", 0F, 300F);
        trany.setInterpolator(new BounceInterpolator());
        trany.setDuration(2500);
        trany.setRepeatCount(0);

    /** --------회전 애니메이션 --------*/
        // X축 회전(360도)
        rx = ObjectAnimator.ofFloat(title, "rotationX", 360);
        rx.setInterpolator(new CycleInterpolator(3f)); // 3회 반복하는 사이클 회전
        rx.setDuration(5000); // 5초 동안 실행
        rx.setRepeatCount(0);
```

```
        // Y축 회전(360도)
        ry = ObjectAnimator.ofFloat(title, "rotationY", 360);
        ry.setInterpolator(new CycleInterpolator(3f));
        ry.setDuration(5000);
        ry.setRepeatCount(0);

        /** -------크기 변화 애니메이션 -------*/
        // X축 확대(기존 크기→ 2배)
        scalex = ObjectAnimator.ofFloat(title, "scaleX", 2f);
        scalex.setInterpolator(new LinearInterpolator()); // 일정한 속도
        scalex.setDuration(2500);
        scalex.setRepeatCount(0);

        // Y축 확대(기존 크기→ 2배)
        scaley = ObjectAnimator.ofFloat(title, "scaleY", 2f);
        scaley.setInterpolator(new LinearInterpolator());
        scaley.setDuration(2500);
        scaley.setRepeatCount(0);

        /** --------투명도 애니메이션 ----------*/
        // Alpha 값(투명도) 변경(0 → 1, 서서히 나타남)
        alpha = ObjectAnimator.ofFloat(title, "alpha", 0.0f, 1.0f);
        alpha.setDuration(2500);
        alpha.setRepeatCount(0);

        // 알파 애니메이션이 끝난 후Y축 회전 시작
        alpha.addListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                super.onAnimationEnd(animation);
                ry.start(); // 알파 애니메이션 끝나면rotationY 실행
            }
        });

    /** ------------------ 버튼 클릭 이벤트 ------------------ */
        btn_start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 버튼 클릭 시 알파 애니메이션부터 시작
                alpha.start();
            }
        });
    }
}
```

위 프로젝트에서는 이동(translation), 회전(rotation), 크기 변경(scale), 흐림(alpha) 애니메이션에 대한 코드를 모두 설정해 두었지만, 실제로 실행되는 동작은 흐림(alpha) 애니메이션이 먼저 실행되고, 그 완료 시점에 맞추어 Y축 회전(ry) 애니메이션만 이어서 동작하도록 구성되어 있습니다.

이 부분에서 특히 주목할 만한 점은 "alpha.addListener(new AnimatorListener Adapter() { ... } " 구문인데, 이는 alpha 애니메이션이 종료된 후 추가로 수행할 동작을 지정하는 기능을 담당합니다. 즉, 하나의 애니메이션 완료 이벤트를 감지하여 그 시점에 맞추어 다른 애니메이션을 연결하는 방식이며, 이는 이전에 학습했던 Property Animation의 .withEndAction() 메서드와 유사한 의미와 역할을 가집니다.

따라서, 위 프로젝트를 직접 실행하여 결과를 확인한 후에는, 현재 alpha → ry 회전으로만 연결된 구조를 확장하여 이동, 회전, 크기 변경 애니메이션도 동일한 방식으로 동작하도록 코드를 변경해 보는 실습을 권장합니다. 예를 들어, tranx 애니메이션이 끝난 후 scalex가 실행되도록 연결하거나, rx 회전 애니메이션이 끝난 뒤 trany 이동 애니메이션이 실행되도록 순차적으로 이어주는 식으로 실험해 볼 수 있습니다. 이 과정에서 addListener()를 활용하면 각 애니메이션이 끝난 직후 다른 애니메이션을 자연스럽게 이어 붙일 수 있으므로, 복합적이고 연속적인 애니메이션 흐름을 구현할 수 있습니다.

이처럼 다양한 애니메이션을 조합해 보고, 종료 시점에 따라 다른 애니메이션을 연계하여 실행하는 과정을 직접 시도해 보면, 단순한 단일 효과를 넘어 조건부, 상태 의존적, 그리고 복합적인 모션 설계 능력을 기를 수 있으며, 안드로이드 애니메이션 시스템의 동작 원리를 더욱 깊이 이해하는 데 도움이 될 것입니다.



연습문제

1. 안드로이드 애니메이션은 View Animation, Drawable Animation, Property Animation으로 구분된다. 각 방식의 특징과 차이점을 설명하고, 실제 앱 개발에서 각각이 적합하게 사용되는 사례를 들어 서술하세요.

2. View Animation(트윈 애니메이션)을 XML 기반으로 구현할 때 사용하는 주요 태그(〈translate〉, 〈scale〉, 〈rotate〉, 〈alpha〉, 〈set〉)의 기능과 활용 방법을 구체적인 예시와 함께 설명하세요.

3. Drawable Animation(프레임 애니메이션)의 동작 원리를 설명하고, 이 방식이 가지는 장점과 단점을 분석하시오. 또한 프레임 애니메이션이 자주 활용되는 실제 예시를 제시하세요.

4. Interpolator의 개념과 역할을 설명하고, LinearInterpolator, AccelerateInter –polator, DecelerateInterpolator, BounceInterpolator를 비교하여 각각의 속도 변화 특징을 서술하세요.

5. Property Animation과 ObjectAnimator의 차이를 설명하시오. 특히 Object –Animator가 View Animation과 달리 속성 값을 직접 변경한다는 점에서 가지는 장점을 중심으로 기술하고, 이러한 특징이 사용자 경험(UX)에 어떤 긍정적인 효과를 주는지 구체적으로 서술하세요.



제6장

액티비티와 인텐트



학습목표

- 액티비티의 생명주기와 상태보존을 설명할 수 있다.
- 명시적, 암시적 인텐트 차이와 구성요소를 설명하고 적절히 구성할 수 있다.
- 백스택, 태스크 동작과 런치 모드 및 대표 플래그의 효과를 분석 및 적용할 수 있다.
- Parcelable을 사용해 경량 데이터 전달을 구현할 수 있다.

6.1 액티비티와 인텐트 기본

6.1.1 액티비티(Activity)

안드로이드에서 Activity(액티비티)란 하나의 독립적인 화면(Screen)을 담당하는 핵심 컴포넌트로, 사용자가 직접 눈으로 확인할 수 있는 UI를 표시하고 버튼 클릭, 입력창 작성, 스크롤과 같은 다양한 사용자 입력을 처리하는 역할을 수행합니다. 쉽게 말해, 앱을 실행했을 때 우리가 마주하는 각 화면이 바로 하나의 액티비티라고 할 수 있으며, 이는 안드로이드 애플리케이션을 구성하는 가장 기본적이고 중요한 단위입니다.

액티비티는 단순히 화면을 보여주는 것에 그치지 않고, LifecycleOwner(생명주기 주체)로서 화면이 생성되고 표시되며, 일시 중지되거나 종료되는 일련의 생명주기를 관리합니다. 이를 통해 시스템 자원을 효율적으로 관리하고, 화면 상태를 적절히 복원하거나 해제할 수 있습니다. 또한 액티비티는 Context 역할을 수행하여 애플리케이션 전반에 걸친 리소스 접근, 인텐트 처리, 권한 요청, 서비스 실행, 브로드캐스트 수신 등 다양한 기능을 지원하며, 앱 운영의 중심 축이라 할 수 있습니다.

하나의 애플리케이션은 보통 여러 개의 액티비티로 구성되며, 각 액티비티는 특정 기능이나 화면을 담당합니다. 예를 들어, 로그인 화면을 담당하는 액티비티, 메인 화면을 담당하는 액티비티, 그리고 설정 화면을 담당하는 액티비티가 각각 따로 존재할 수 있습니다. 이러한 액티비티 간에는 인텐트(Intent)라는 메시지 객체를 통해 서로 화면 전환을 하거나 데이터를 주고받을 수 있으며, 이를 통해 사용자에게 유기적이고 일관된 앱 사용 경험을 제공합니다.

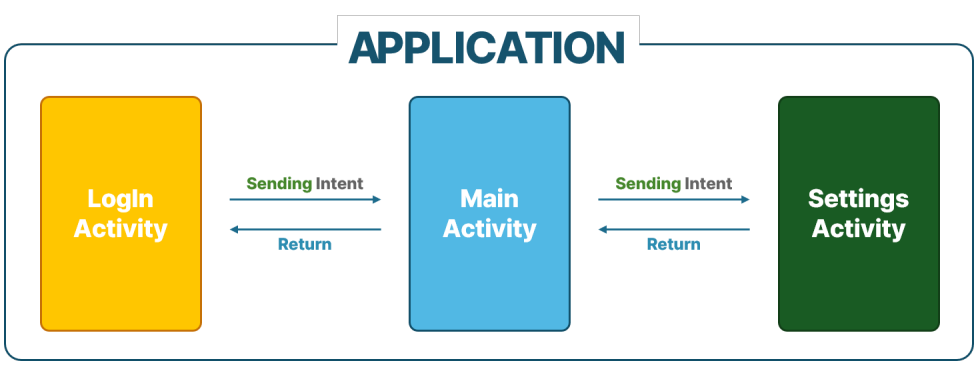


그림 6-1 인텐트 예시

정리하면, 액티비티는 안드로이드 애플리케이션의 뼈대를 이루는 단위이자 UI 표시와 사용자 상호작용의 중심이며, 동시에 생명주기와 컨텍스트를 관리하는 중요한 컴포넌트로서 앱 개발에 있어 반드시 이해하고 다루어야 할 핵심 요소라고 할 수 있습니다.

(1) 액티비티 생명주기(Lifecycle)

안드로이드의 액티비티(Activity) 생명주기(Activity Lifecycle) 는 앱 화면이 생성되고, 사용자와 상호작용하며, 종료되기까지의 전체 흐름을 정의하는 중요한 개념입니다. 이를 이해하면 효율적인 리소스 관리와 안정적인 앱 동작을 구현할 수 있습니다.

아래는 액티비티 생명주기 주요 콜백 메서드입니다.

표 6-1 액티비티 생명주기 주요 콜백 메서드

상태	대표 콜백	설명
생성	onCreate()	최초 초기화(뷰 바인딩, DI, ViewModel연결)
시작	onStart()	화면이 보이기 시작, UI 갱신 준비
재개	onResume()	포그라운드, 사용자 상호작용 가능(애니메이션/카메라 시작)
일시중지	onPause()	포커스 상실 직전, 매우 짧음 시간
정지	onStop()	화면이 완전히 가려짐
재시작	onRestart()	정지→다시 시작될 때 호출
소멸	onDestroy()	최종 정리(바인딩 해제). 시스템에 의해 예고없이 종료될 수 있음

(2) 인텐트(Intent), 태스크, 백스택, 런치 모드(Launch Modes)

❶ 인텐트

다음 섹션에서 인텐트(Intent)의 동작 방식과 활용 방법에 대해 보다 자세히 살펴보기로 하고, 여기에서는 우선 인텐트의 기본적인 개념만 간단히 설명하도록 하겠습니다. 인텐트는 안드로이드에서 컴포넌트 간 통신을 담당하는 핵심 메커니즘으로, 화면 간 전환이나 데이터 전달, 시스템 기능 호출 등 다양한 역할을 수행합니다. 크게 구분하면 명시적 인텐트(Explicit Intent) 와 암시적 인텐트(Implicit Intent) 두 가지 형태가 있으며, 명시적 인텐트는 특정 컴포넌트를 명확히 지정하여 호출하는 방식이고, 암시적 인텐트는 수행할 작업의 성격만을 정의하여 안드로이드 시스템이 해당 작업을 처리할 수 있는 적절한 컴포넌트를 찾아 실행하는 방식입니다.

- 명시적 인텐트 : 특정 액티비티로 이동

```
//명시적 인텐트 예시
Intent i = new Intent(this, DetailActivity.class);
i.putExtra("id", 123);
startActivity(i);
```

- 암시적 인텐트 : 작업에 맞는 외부 액티비티 호출

```
//암시적 인텐트 예시
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://example.com")));
```

❷ 태스크, 백스택

안드로이드에서 액티비티(Activity)는 스택(Stack) 구조로 관리되며, 사용자가 새로운 액티비티를 실행할 때마다 이전 액티비티 위에 차례대로 쌓이게 됩니다. 이러한 구조는 사용자가 화면을 이동할 때 직관적인 흐름을 제공하며, 뒤로 가기 버튼을 눌렀을 때 가장 최근에 열었던 화면부터 순차적으로 닫히도록 합니다.

이와 관련하여 두 가지 중요한 개념이 있습니다.

- Task: 여러 액티비티들이 모여 하나의 실행 단위를 형성한 것을 의미합니다. 즉, 하나의 태스크(Task)는 액티비티들의 스택 모음이며, 사용자가 최근 사용 앱 화면에서 확인할 수 있는 항목 하나가 보통 하나의 태스크에 해당합니다. 예를 들어, 사용자가 메신저 앱을 실행하면 그 안에서 채팅방, 프로필 설정, 사진 보기 등의 액티비티들이 차례대로 쌓여 하나의 태스크를 구성하게 됩니다.
- Back Stack: 스택에 쌓여 있는 액티비티들의 구조를 가리키며, 가장 최근에 실행된 액티비티가 스택의 맨 위에 위치합니다. 사용자가 뒤로 가기(Back) 버튼을 누르면, 맨 위에 있는 액티비티가 pop 되어 사라지고, 그 아래에 있던 이전 액티비티가 다시 화면에 나타납니다. 이러한 방식 덕분에 앱은 사용자의 탐색 흐름을 자연스럽게 유지할 수 있습니다.

즉, 액티비티와 태스크(Task), 그리고 백스택(Back Stack) 구조는 앱의 화면 전환과 사용자 경험을 체계적으로 관리하는 핵심 메커니즘이라고 할 수 있습니다.

❸ 런치(Launch) 모드(매니페스트, manifest)

안드로이드에서 런치 모드(Launch Mode)란, app 〉 manifests 〉 AndroidManifest.xml 파일 내의 〈activity〉 태그에서 지정할 수 있는 속성으로, 특정 액티비티가 실행될 때 그 인스턴스를 새로 생성할지, 이미 존재하는 인스턴스를 재활용할지, 그리고 해당 액티비티가 태스크(Task)와 백스택(Back Stack) 내에서 어떤 방식으로 배치될지를 결정하는 중요한 설정입니다.

즉, 런치 모드는 단순히 액티비티가 열리는 방식을 제어하는 것이 아니라, 앱 전체의 화면 전환 흐름과 태스크 관리 정책을 좌우하는 요소라고 할 수 있습니다. 이를 통해 개발자는 액티비티의 실행 방식과 스택 구조를 원하는 대로 제어할 수 있으며, 동일한 액티비티가 불필요하게 여러 번 생성되는 것을 방지하거나, 특정 상황에서는 항상 새로운 인스턴스를 생성하여 일관된 동작을 보장할 수도 있습니다.

이처럼 런치 모드 설정은 사용자 경험(UX)을 고려한 앱 설계에 큰 영향을 미치며, 적절히 활용하면 액티비티 간 이동을 효율적으로 관리하고 앱의 안정성과 일관성을 높일 수 있습니다.

```
// AndroidManifest.xml 예시
<activity
    android:name=".SomeActivity"
    android:launchMode="standard|singleTop|singleTask|singleInstance" />
```

표 6-2 Launch Mode

모드	인스턴스 생성	재호출시 동작	콜백 포인트	테스크 특성
standard (기본)	호출마다 새 인스턴스	스택 위에 계속 push	평소처럼 onCreate()	일반 BackStack
singleTop	최상단이 동일하면 재사용	최상단이면 onNewIntent()	onNewIntent()	일반 BackStack
sigleTask	Task 내 1개만 존재	기존에 있으면 그 태스크 앞으로 bring, 위의 액티비티를 제거	대상에 onNewIntent()	Task의 루트/핵심 화면용
sigleInstance	전용 Task에 오직 1개만	항상 독립 Task	onNewIntent()	분리된 전용Task

❹ Intent Flag

[표 6-2]는 AndroidManifest.xml에서 런치 모드를 지정하는 방법을 보여주고 있는데, 이는 애플리케이션 실행 시 고정된 형태로 동작 방식을 설정할 때 주로 사용됩니다. 그러나 실제 개발 현장에서는 앱의 실행 흐름이 항상 동일하지 않고, 상황이나 조건에 따라 유동적으로 화면 전환 방식이나 액티비티의 생성을 제어해야 하는 경우가 많습니다.

따라서 AndroidManifest.xml에서 미리 고정하는 방식보다는, MainActivity.java와 같은 자바 코드 파일에서 직접 런치(Launch) 모드를 제어하는 방법이 더 자주 활용됩니다. 이렇게 하면 실행 시점에 따라 적절한 런치 모드를 선택하여 적용할 수 있으므로, 다양한 사용자 시나리오에 대응할 수 있는 유연한 앱 설계가 가능합니다.

아래의 [표 6-3]은 이러한 코드 기반 설정에서 활용되는 런치(Launch) 모드 관련 플래그(Flags)를 정리한 것으로, 실제로 Intent 객체에 플래그를 지정하여 액티비티가 생성되거나 재사용되는 방식을 세밀하게 제어할 수 있습니다. 이를 통해 개발자는 스택(Stack) 관리, 새로운 태스크(Task) 생성, 기존 액티비티 재활용 여부 등을 코드 레벨에서 명확히 제어할 수 있으며, 복잡한 화면 전환 로직이나 특수한 사용자 흐름을 보다 효과적으로 구현할 수 있게 됩니다.

표 6-3 Intent flag

플래그	효과	비고
FLAG_ACTIVITY_SIGLE_TOP	최상단이 동일하면 재사용	sigleTop과 유사
FLAG_ACTIVITY_CLEAR_TOP	목표 액티비티 위의 스택을 제거	sigleTask의 효과와 유사0
FLAG_ACTIVITY_NEW_TASK	새 태스크에서 실행 (또는 기존 태스크 앞으로)	암시적 인텐트 / 알림에서 자주 사용
FLAG_ACTIVITY_MULTIPLE_TASK	강제 다중 태스크(특수)	대부분 비권장

```
// Intent Flag 적용 예시
Intent i = new Intent(this, MainActivity.class);
i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_SINGLE_TOP);
startActivity(i);
// → MainActivity가 위로 올라오고, 위의 액티비티는 제거. Main 최상단이면 onNewIntent().
```

❺ Task Affinity(Task 관계)

액티비티는 실행될 때 반드시 하나의 태스크(Task)에 속하게 되며, 어떤 태스크에 포함될 것인지는 시스템이 정하거나 개발자가 명시적으로 제어할 수 있습니다. 이때 액티비티가 어떤 태

스크에 속할지를 지정하는 일종의 선호 그룹(preferred group) 개념이 존재하며, 이는 런치 모드(Launch Mode) 중 특히 singleTask와 singleInstance 모드의 동작 방식에 직접적인 영향을 줍니다.

예를 들어, singleTask 모드는 특정 액티비티가 하나의 태스크 안에서 유일하게 존재하도록 강제하기 때문에, 새로운 인스턴스를 생성하는 대신 기존에 존재하는 액티비티를 재활용하게 됩니다. 이때 액티비티가 어떤 태스크에 속할지에 대한 선호 그룹 설정이 반영되어, 해당 그룹 내 태스크가 우선적으로 사용됩니다. 반면, singleInstance 모드는 그 액티비티가 아예 독립된 태스크를 차지하도록 하여, 다른 액티비티와 섞이지 않고 단독으로 실행되게 합니다. 따라서 선호 그룹 개념은 이러한 런치 모드가 적용될 때, 액티비티 인스턴스가 생성되는 위치와 태스크의 구조를 결정하는 중요한 기준이 됩니다.

즉, 액티비티의 태스크 소속을 제어하는 이 개념은 단순히 실행 방식에 영향을 주는 것에 그치지 않고, 백스택 관리, 화면 전환 흐름, 그리고 사용자 경험(UX)의 일관성에도 직결되는 핵심 요소라 할 수 있습니다.

```
// task Affinity 적용 예시
<activity
    android:name=".ReaderActivity"
    android:launchMode="singleTask"
    android:taskAffinity="com.example.reader" /> <!-- 별도 태스크 성향 -->
```

단, 액티비티의 Affinity를 지나치게 세분화하거나 무분별하게 나누게 되면 의도치 않은 부작용이 발생할 수 있습니다. 특히, 안드로이드 시스템에서 태스크(Task) 단위로 최근 사용한 앱 화면을 표시할 때, 동일한 애플리케이션임에도 불구하고 여러 개의 태스크가 각각 독립된 카드 형태로 나타나게 됩니다. 이렇게 되면 사용자는 동일한 앱이 여러 카드로 분리되어 보이는 상황을 겪게 되고, 앱의 실행 흐름을 직관적으로 이해하거나 다시 돌아가는 과정이 불필요하게 복잡해질 수 있습니다. 결국 이는 사용자 경험(UX)을 저하시켜 앱 사용의 일관성과 편의성을 해치게 되므로, Affinity 설정은 반드시 필요한 경우에만 신중하게 적용하는 것이 바람직합니다.

6.1.2 인텐트 (Intent)

안드로이드에서 매우 핵심적인 개념으로, 앱 컴포넌트(Activity, Service, Broadcast Receiver 등)에게 특정한 작업을 요청하는 역할을 하는 일종의 메시지 객체입니다. 인텐트는 단순히 앱 내부에서 화면을 전환하기 위한 수단으로만 사용되는 것이 아니라, 백그라운드 작업 실행, 브로드캐

스트 메시지 전송, 그리고 애플리케이션 간 상호 작용을 가능하게 하는 강력한 도구로 활용됩니다. 예를 들어, 사용자가 버튼을 눌러 새로운 화면으로 이동하는 경우에는 액티비티 간 화면 전환을 위해 인텐트를 사용하고, 음악을 재생하거나 다운로드와 같은 작업을 백그라운드에서 처리할 때는 서비스(Service)를 호출하기 위해 인텐트가 사용됩니다. 또한, 브로드캐스트 리시버(BroadcastReceiver)를 통해 시스템이나 다른 앱에서 발생한 이벤트를 전달할 때에도 인텐트가 중요한 역할을 담당합니다.

특히, 인텐트는 앱과 앱 사이의 경계를 넘는 상호작용을 가능하게 합니다. 예를 들어, 내 앱에서 촬영한 사진을 다른 앱에 공유하거나, 특정 주소를 지도 앱에서 열거나, URL을 브라우저로 연결하는 등의 기능은 모두 인텐트를 통해 이루어집니다. 이처럼 인텐트는 앱 내부의 컴포넌트들을 연결할 뿐만 아니라, 외부 애플리케이션과도 자연스럽게 연동할 수 있도록 함으로써 사용자에게 더 풍부하고 통합적인 경험을 제공합니다.

(1) 인텐트 구성 요소

인텐트(Intent)는 단순히 "무언가를 실행하라"는 요청을 담는 메시지 객체에 그치지 않고, 다양한 구성요소(Component)들을 통해 세부적으로 어떤 작업을 수행할지, 어떤 데이터와 함께 전달할지, 또 어떤 방식으로 실행될지를 결정할 수 있습니다. 인텐트는 이러한 요소들이 모여 하나의 구조를 이루며, 이를 통해 안드로이드 시스템은 요청을 해석하고 적절한 컴포넌트를 실행할 수 있게 됩니다. 인텐트를 구성하는 주요 요소들은 아래와 같습니다.

- Component : 인텐트가 실행해야 할 대상 컴포넌트를 직접 지정하는 요소입니다. 예를 들어 특정 액티비티 클래스 이름을 명확하게 지정하면 이를 명시적 인텐트 (Explicit Intent)라 하며, 지정하지 않고 시스템이 적절한 컴포넌트를 찾아 실행하게 하면 암시적 인텐트(Implicit Intent)가 됩니다.
- Action: 수행하고자 하는 동작을 정의하는 문자열 값입니다. 예를 들어 웹 페이지를 열고 싶을 때는 ACTION_VIEW, 다른 앱에 데이터를 전송하고 싶을 때는 ACTION_SEND와 같은 액션을 사용합니다.
- Data(Uri) & Type(MIME): 인텐트가 동작할 대상과 데이터의 형식을 지정하는 요소입니다. 예를 들어 Uri를 통해 특정 웹 주소나 전화번호를 지정할 수 있고, MIME 타입을 통해 데이터의 종류(예: 이미지, 텍스트, 오디오)를 명시할 수 있습니다.

- Category : 인텐트에 추가적인 속성을 부여하는 역할을 합니다. 예를 들어 CATEGORY_LAUNCHER는 런처 화면에 표시될 수 있는 액티비티임을 의미합니다.
- Extras : Bundle 형태로 전달되는 부가 데이터입니다. 다른 앱과 상호작용하거나 특정 화면으로 이동할 때 필요한 정보를 담아 전달할 수 있습니다. 예를 들어 로그인 화면에서 입력한 사용자 이름이나 메시지 작성 시의 텍스트 내용을 담아 전달할 수 있습니다.
- Flags : 액티비티의 실행 방식이나 태스크(Task) 및 백스택(Stack) 관리 방식, 권한 관련 동작 등을 제어하는 데 사용됩니다. 이를 통해 액티비티가 새로운 태스크에서 실행될지, 기존 액티비티를 재사용할지 등을 세밀하게 조정할 수 있습니다.
- Package / ClipData : 특정 패키지 이름을 지정하여 인텐트가 실행될 앱을 제한하거나, 복수의 데이터를 전달할 때 ClipData를 활용할 수 있습니다. 이를 통해 데이터 공유와 앱 간 통신을 더욱 정밀하게 제어할 수 있습니다.

인텐트는 단순한 요청 메시지가 아니라, 수행할 동작(Action), 다룰 데이터(Data), 실행 대상(Component), 실행 방식(Flags) 등 다양한 요소들이 종합적으로 정의된 구조체라고 할 수 있습니다. 이를 통해 안드로이드 앱은 내부적으로 화면 전환을 관리하고, 외부 앱과도 유기적으로 연결되며, 사용자에게 자연스럽고 직관적인 앱 사용 경험을 제공합니다.

(2) 인텐트 종류 및 주 용도

안드로이드에서 인텐트(Intent)는 크게 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent)로 구분되며, 활용 목적에 따라 브로드캐스트용 인텐트, 서비스용 인텐트, 그리고 PendingIntent 등으로 세분화할 수 있습니다.

표 6-4 Intent 종료 및 사용 용도

종류	설명	대표 메서드
명시적 (Explicit)	대상 컴포넌트를 정확히 지정	startActivity(new Intent(this, 클래스명)
암시적 (Implicit)	동작/데이터만 제시 →시스템이 적합한 대상 해결	startActivity(new Intent(ACTION_VIEW, uri))
브로드캐스트	여러 수신자에게 이벤트 방송	sendBroadcast(intent)
서비스용	백그라운드 작업 시작 / 바인딩	startService(), bindService()
PendingIntent	현재 Intente 대신에 실행 요청(미래)으로 시스템에게 지연실행 토큰(권한 위임핸들)	알림(Notification), 알람(AlarmManager)

(3) 인텐트 필터와 매칭 규칙

수신자(주로 액티비티)는 매니페스트(AndroidManifest.xml)의 〈intent-filter〉 태그를 통해 자신이 처리할 수 있는 인텐트를 명시적으로 선언하게 됩니다. 이는 특히 암시적 인텐트(Implicit Intent) 동작에 있어 핵심적인 요소로, 시스템이 해당 인텐트를 처리할 수 있는 적절한 컴포넌트를 결정하는 기준이 됩니다. 하지만 이 과정에서 요청된 인텐트와 필터 조건이 하나라도 맞지 않으면 해당 컴포넌트는 실행 대상이 될 수 없습니다. 구체적인 매칭 조건은 다음과 같습니다.

- Action: 요청된 인텐트의 액션 값이 수신자의 인텐트 필터에 선언된 액션 값과 하나 이상 일치해야 합니다.
- Category: 요청된 인텐트가 가지고 있는 모든 카테고리가 인텐트 필터에 반드시 포함되어야 합니다. 하나라도 누락되면 매칭되지 않습니다.
- Data(Type/URI): 요청된 인텐트에 포함된 MIME 타입, 스킴(Scheme), 호스트(Host), 패스(Path) 등이 필터에서 정의된 조건을 모두 충족해야 합니다.

만약 이러한 조건들이 맞지 않아 실행 가능한 액티비티가 존재하지 않는 경우, 앱은 실행 중에 ActivityNotFoundException 예외가 발생할 수 있습니다. 따라서 안전한 실행을 위해 코드 레벨에서 반드시 이 예외 상황을 대비한 예외 처리 규칙을 두어야 하며, 이를 통해 앱이 갑작스럽게 종료되는 문제를 방지하고 사용자에게 적절한 대체 동작이나 안내 메시지를 제공할 수 있습니다.

(4) Action, Category, Data

인텐트(Intent)를 사용할 때는 여러 구성 요소 중에서도 Action(동작 지정), Category(추가 분류 정보), 그리고 Data(연결할 데이터와 그 타입)가 가장 핵심적으로 활용됩니다. 이 세 가지 요소는 인텐트가 어떤 작업을 수행해야 하는지, 어떤 맥락에서 실행되어야 하는지, 그리고 어떤 데이터에 대해 동작해야 하는지를 명확히 정의해 주는 역할을 합니다.

우선 Action은 인텐트가 수행할 구체적인 동작을 나타내며, 예를 들어 웹 페이지를 열 때는 ACTION_VIEW, 다른 앱으로 데이터를 전송할 때는 ACTION_SEND와 같은 값이 사용됩니다. Category는 인텐트가 실행되는 추가적인 맥락이나 성격을 부여하는데 활용되며, 예를 들어 CATEGORY_DEFAULT나 CATEGORY_BROWSABLE과 같은 카테고리를 통해 인텐트의 실행 환경을 구체화할 수 있습니다. 마지막으로 Data는 인텐트가 처리해야 하는 실제 대상을 지정하는 요소로, URI와 MIME 타입을 함께 정의하여 어떤 종류의 데이터(예: 이미지, 텍스트, 웹

주소 등)를 어떤 방식으로 처리할 것인지 명확히 전달합니다.

따라서 Action, Category, Data는 서로 결합되어 인텐트의 의미와 실행 조건을 결정하는 중요한 요소들이며, 이를 올바르게 설정해야 원하는 컴포넌트가 정확히 실행되고 의도한 동작이 정상적으로 이루어질 수 있습니다.

- Action:
  - Intent.ACTION_VIEW(보기), ACTION_SEND(공유), ACTION_DIAL / ACTION_CALL(전화), ACTION_SENDTO(문자/메일), ACTION_GET_CONTENT(단건 선택), ACTION_PICK(선택), ACTION_EDIT 등
- Category:
  - CATEGORY_DEFAULT(기본), CATEGORY_BROWSABLE(브라우저 / 딥링크 허용)
- Data(Type/URI):
  - Uri.parse("http://..."), Uri.parse("geo:37.5, 127.0"), content://
  - MIME: "text/plain", "image/*", "application/pdf"

(5) 사용예시

❶ 명시적 인텐트: 화면 전화 + 데이터 전달

```
// 송신측 Intent
Intent i = new Intent(this, DetailActivity.class);
i.putExtra("id", 123);  // id라는 변수에 123 이라는 값을 담아둠
startActivity(i);

// 수신측 Intentt
int id = getIntent().getIntExtra("id", -1);
// 송신측 Intent에서 넘겨준 id 값(123)을 수신측 변수 id에 숫자형으로 받음
// 만약, 송신측에서 보내온 id 에 값이 없을 경우 default 값 -1 로 대치
```

❷ 암시적 인텐트: 웹 / 지도 / 전화 / 공유

```
// 웹 열기
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://example.com")));

// 지도(좌표)
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("geo:37.5665,126.9780")));
```

```
// 전화 앱 열기(다이얼 화면)
startActivity(new Intent(Intent.ACTION_DIAL, Uri.parse("tel:01012345678")));

// 텍스트 공유
Intent share = new Intent(Intent.ACTION_SEND);
share.setType("text/plain");
share.putExtra(Intent.EXTRA_TEXT, "안녕하세요!");
startActivity(Intent.createChooser(share, "공유하기"));
```

❸ 콘텐츠 선택

```
Intent pick = new Intent(Intent.ACTION_GET_CONTENT);
pick.setType("image/*");
startActivity(pick); // Activity Result API와 함께 쓰면 더 좋음
```

❹ FileProvider로 이미지 공유

```
<!-- AndroidManifest.xml -->
<provider
  android:name="androidx.core.content.FileProvider"
  android:authorities="${applicationId}.fileprovider"
  android:exported="false"
  android:grantUriPermissions="true">
  <meta-data
    android:name="android.support.FILE_PROVIDER_PATHS"
    android:resource="@xml/file_paths"/>
</provider>
```

```
<!-- res/xml/file_paths.xml -->
<paths>
  <external-files-path name="images" path="Pictures/"/>
</paths>
```

```
//MainActivity.java
Uri uri = FileProvider.getUriForFile(this, getPackageName()+".fileprovider",
imageFile);
Intent send = new Intent(Intent.ACTION_SEND);
send.setType("image/*");
send.putExtra(Intent.EXTRA_STREAM, uri);
send.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
startActivity(Intent.createChooser(send, "이미지 공유"));
```

❺ 브로드캐스트 전송 / 수신

```
// 전송
sendBroadcast(new Intent("com.example.ACTION_SYNC_REQUEST"));

// 수신자(런타임 등록)
BroadcastReceiver r = new BroadcastReceiver() {
  @Override public void onReceive(Context c, Intent i) {
    // 처리
  }
};
registerReceiver(r, new IntentFilter("com.example.ACTION_SYNC_REQUEST"));
// onStop 등에서 unregisterReceiver(r);
```

❻ 서비스 시작 / 바인딩

```
// 시작
startService(new Intent(this, SyncService.class));
// 바인딩
bindService(new Intent(this, SyncService.class), conn, BIND_AUTO_CREATE);
```

❼ PendingIntent(알림 탭 시 화면열기)

```
Intent open = new Intent(this, DetailActivity.class).putExtra("id", 7);
PendingIntent pi = PendingIntent.getActivity(this, 0, open,
        PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE);
// Notification에 setContentIntent(pi)
```

6.2 액티비티와 인텐트 실습 예제

6.2.1 명시적 인텐트: 화면 전환 + parcelable 전달

명시적 Intent 실습을 위해 아래의 설정에 따라 프로젝트를 생성하세요.

- Project Name : screenExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) 액티비티 생성

❶ 모델 파일 생성 (Parcelable)

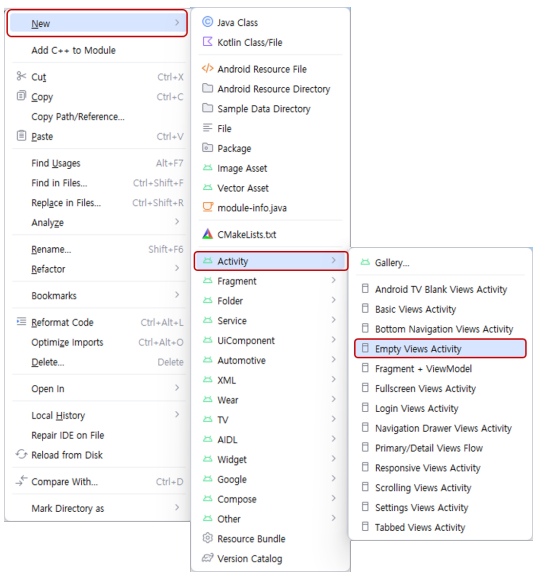
app 〉 java 〉 com.example.screenexam 폴더 선택 후 우클릭하고, new 〉 Java Class 클릭하세요. 아래와 같이 파일명을 user 라고 입력하세요.

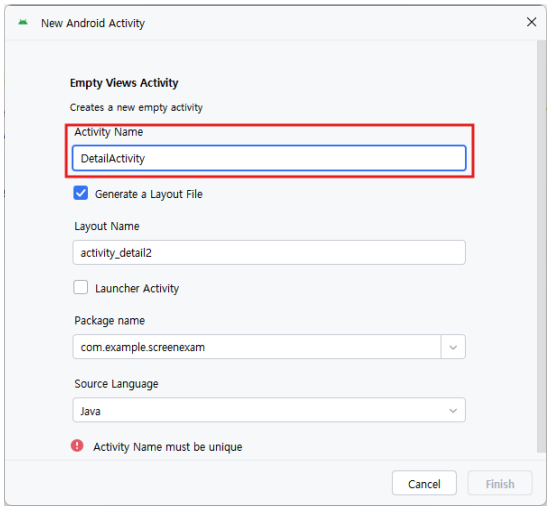


❷ 추가 Activity 생성

액티비티와 해당 액티비티 레이아웃(XML 디자인)이 함께 생성하여야 Manifest에 추가되는 Activity가 등록됩니다.

app 〉 java 〉 com.example.screenexam 폴더에서 우클릭 후, New 〉 Activity 〉 Empty Views Activity를 선택하고, DetailActivity를 입력하세요.





위 그림에서 Activity Name에 대한 이름만 입력하면 Layout Name은 자동 입력되니 따로 입력하지 않아도 됩니다.

아래 그림은 추가 Activity 파일을 생성한 후 추가된 Activity가 등록된 메니페스트 파일을 확인할 수 있습니다.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="screenExam"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.ScreenExam">
        <activity
            android:name=".DetailActivity"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

(2) Layout XML

❶ 메인화면 Layout 디자인

[Code] 모드로 변환한 후 아래 코드를 입력하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DetailActivity">

    <Button
        android:id="@+id/btn_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="인텐트"
        android:textSize="20dp" />
</RelativeLayout>
```

❷ 추가화면 XML 디자인

앞 단계에서 자동생성된 activity_detail.xml 파일을 [Code] 모드로 변경 후 아래와 같이 입력하세요.

activity_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tvInfo"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:textSize="20dp" />

    </RelativeLayout>
```

❸ 코드 작성

User.java

```
    package com.example.screenexam;

    import android.os.Parcel;
    import android.os.Parcelable;

    // User 클래스는Parcelable 인터페이스를 구현하여
    // 객체를Intent에 담아 다른Activity로 전달할 수 있도록 함
    public class User implements Parcelable {

        // 사용자 이름과 나이 필드
        public String name;
        public int age;

        // 일반 생성자
        public User(String n, int a) {
            name = n;
            age = a;
        }

        // Parcel로부터 데이터를 읽어와User 객체를 생성하는 생성자
        protected User(Parcel in) {
            name = in.readString(); // Parcel에서name 읽기
            age = in.readInt();     // Parcel에서age 읽기
        }

        // Parcelable.Creator: User 객체를 생성하거나 배열을 만들 때 사용
        public static final Creator<User> CREATOR = new Creator<User>() {

            // Parcel로부터User 객체 생성
            public User createFromParcel(Parcel in) {
                return new User(in);
            }

            // User 객체 배열 생성
            public User[] newArray(int size) {
```

```
            return new User[size];
        }
    };

    // 특별한 내용 설명 필요 없으므로0 반환(파일 디스크립터 등은 사용하지 않음)
    public int describeContents() {
        return 0;
    }

    // 객체 데이터를Parcel에 쓰는 메서드(역직렬화용)
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name); // name 쓰기
        dest.writeInt(age);     // age 쓰기
    }
}
```

MainActivity.java (송신측)

```
package com.example.screenexam;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        Button btn_s = (Button) findViewById(R.id.btn_start);

        // 버튼 클릭 시 실행될 이벤트 리스너 등록
        btn_s.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // DetailActivity 화면으로 전환하기 위한Intent 생성
                // (현재Context → DetailActivity 지정)
                Intent i = new Intent(getApplicationContext(), DetailActivity.class);

                // 새로운User 객체 생성 후Intent에 추가
                // Parcelable을 구현했기 때문에putExtra로 전달 가능
                i.putExtra("user", new User("Alice", 25));
```

```
            // DetailActivity 실행
            startActivity(i);
        }
    });
  }
}
```

DetailActivity.java (수신측)

```
package com.example.screenexam;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class DetailActivity extends AppCompatActivity {

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

        // MainActivity에서 전달된Intent로부터 "user" 객체 꺼내오기
        // User 클래스가Parcelable을 구현했기 때문에 직렬화/역직렬화 가능
        User user = getIntent().getParcelableExtra("user");

        // 레이아웃에 정의된TextView 가져오기
        TextView tv = (TextView) findViewById(R.id.tvInfo);

        // TextView에User 객체의 이름과 나이 표시
        tv.setText("Name=" + user.name + ", Age=" + user.age);
    }
}
```

6.2.2 암시적 인텐트 (웹 / 통화 / 지도 / 공유)

암시적 Intent 실습을 위해 아래의 설정에 따라 프로젝트를 생성하세요.

- Project Name : implicitExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

해당 프로젝트에서 실행할 인텐트는 모두 안드로이드 기본 어플을 호출하기 때문에 명시적 인텐트와 같이 여러 개의 파일을 생성하지 않아도 됩니다.

(1) 레이아웃(XML) 디자인

[Code] 모드로 변환하고, 아래와 같이 입력하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_web"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_marginStart="20dp"
        android:text="웹"
        android:textSize="20dp" />

    <Button
        android:id="@+id/btn_Dial"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_marginStart="120dp"
        android:text="전화" />

    <Button
        android:id="@+id/btn_Geo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_marginStart="220dp"
        android:text="지도" />

    <Button
        android:id="@+id/btn_share"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_marginStart="320dp"
            android:text="공유" />
    </RelativeLayout>
```

(2) 코드 작성

MainActivity.java

```
    package com.example.implicitexam;

    import android.content.Intent;
    import android.net.Uri;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;
    import androidx.appcompat.app.AppCompatActivity;

    public class MainActivity extends AppCompatActivity {

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

            Button btn_w = (Button) findViewById(R.id.btn_web);
            Button btn_t = (Button) findViewById(R.id.btn_Dial);
            Button btn_g = (Button) findViewById(R.id.btn_Geo);
            Button btn_s = (Button) findViewById(R.id.btn_share);

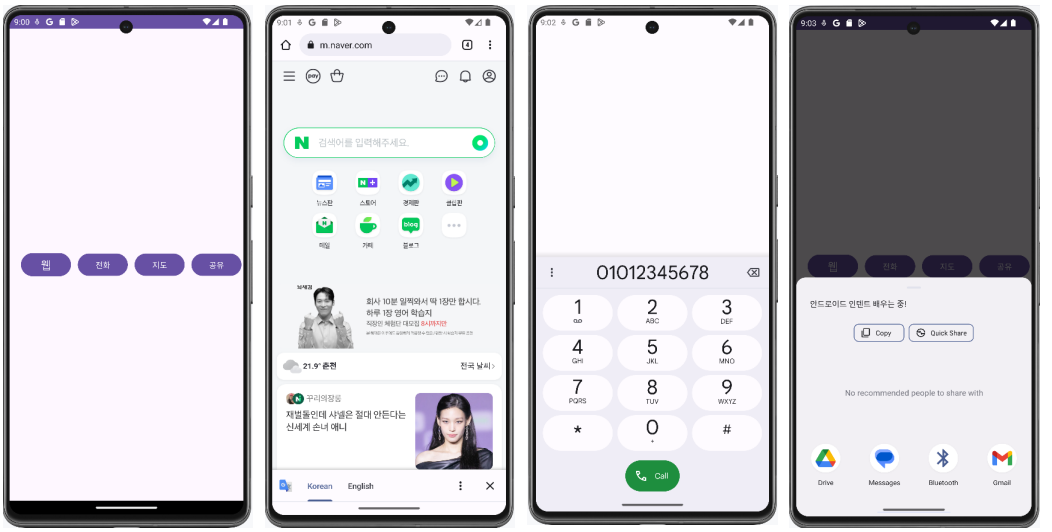
            btn_w.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                // 웹 열기
                startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("https://naver.com")));
                }
            });

            btn_t.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 다이얼
                  startActivity(new Intent(Intent.ACTION_DIAL, Uri.parse("tel:01012345678")));
```

```
            }
        });

        btn_g.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 지도 좌표
                startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("geo:37.5665,126.9780")));
            }
        });

        btn_s.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 텍스트 공유
                Intent share = new Intent(Intent.ACTION_SEND);
                share.setType("text/plain");
                share.putExtra(Intent.EXTRA_TEXT, "안드로이드 인텐트 배우는 중!");
                startActivity(Intent.createChooser(share, "공유하기"));
            }
        });
    }
}
```



6.2.3 카메라

카메라와 같은 안드로이드 기기의 자원이 필요할 때는 권한이 반드시 지정되어야 합니다. 물론 이러한 권한을 등록하기 위해서는 메니페스트(Manifest.xml) 파일을 통해 등록하면 된다.

실습을 위해 아래의 설정에 따라 프로젝트를 생성하세요.

- Project Name : cameraExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) 카메라 권한 설정

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />
    <uses-permission android:name="android.permission.CAMERA" />

<!-- 이하 생략 -->
```

(2)레이아웃 디자인

android_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btn_camera"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:layout_centerInParent="true"
            android:text="카메라"
            android:textSize="20dp" />
    </RelativeLayout>
```

(3) 코드 작성

MainActivity.java

```
    package com.example.cameraexam;

    import android.Manifest;
    import android.content.Intent;
    import android.os.Bundle;
    import android.provider.MediaStore;
    import android.view.View;
    import android.widget.Button;
    import android.widget.Toast;

    import androidx.activity.result.ActivityResultLauncher;
    import androidx.activity.result.contract.ActivityResultContracts;
    import androidx.appcompat.app.AppCompatActivity;

    public class MainActivity extends AppCompatActivity {

        // 카메라 권한 요청을 처리하기 위한 ActivityResultLauncher
        private ActivityResultLauncher<String> requestCamera;

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

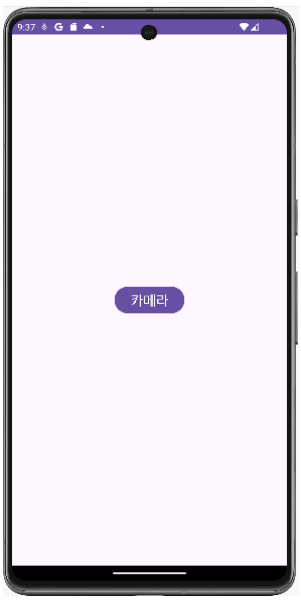
            Button btn_camera = (Button) findViewById(R.id.btn_camera);

            // 권한 요청 결과를 처리할ActivityResultLauncher 등록
            requestCamera = registerForActivityResult(
                            new ActivityResultContracts.RequestPermission(), // 단일 권한 요청
                             granted -> Toast.makeText(getApplicationContext(),
                            "CAMERA: " + granted, Toast.LENGTH_LONG).show()
            ); // granted 결과 값이 -> true(카메라 권한 허용상태), false(거부상태)

            // 버튼 클릭 이벤트 리스너 등록
            btn_camera.setOnClickListener(new View.OnClickListener() {
                @Override
```

```
            public void onClick(View v) {
                // CAMERA 권한 요청 실행
                requestCamera.launch(Manifest.permission.CAMERA);

                // 카메라 앱 실행 인텐트(기본 카메라 앱 열기)
                startActivity(new Intent(MediaStore.ACTION_IMAGE_CAPTURE));
            }
        });
    }
}
```





연습문제

1. 액티비티 생명주기(Activity Lifecycle)의 주요 콜백 메서드(onCreate, onStart, onResume, onPause, onStop, onRestart, onDestroy)의 실행 시점과 역할을 각각 설명하고, 앱의 안정적인 동작과 리소스 관리에서 이를 이해하는 것이 왜 중요한지 서술하세요.

2. 명시적 인텐트(Explicit Intent)와 암시적 인텐트(Implicit Intent)의 개념과 차이를 설명하고, 각각이 사용되는 실제 예시(예: 내부 액티비티 전환, 외부 앱 호출)를 들어 비교하여 서술하세요.

3. 안드로이드에서의 Task와 Back Stack 구조가 사용자 경험(UX)에 어떤 영향을 주는지 설명하고, '뒤로 가기' 버튼 동작과의 연관성을 중심으로 서술하세요.

4. 런치 모드(Launch Mode)와 Intent Flag는 액티비티의 실행 및 스택 관리 방식에 직접적인 영향을 준다. 주요 런치 모드(standard, singleTop, singleTask, singleInstance)와 대표적인 플래그(FLAG_ACTIVITY_SINGLE_TOP, FLAG_ACTI VITY_CLEAR_TOP, FLAG_ACTIVITY_NEW_TASK)의 특징과 동작 방식을 비교하여 서술하세요.

5. 안드로이드 인텐트의 주요 구성 요소인 Action, Data, Category, Extras, Flags의 역할을 설명하고, 이들이 결합하여 인텐트 실행 대상을 결정하는 과정을 구체적인 예시와 함께 서술하세요.



제 7 장

어댑터 뷰



학습목표

- ArrayAdapter, SimpleAdapter, CursorAdapter, BaseAdapter를 구분하여 적용할 수 있다.
- ListView를 활용하여 프로그램을 작성할 수 있다.
- ArrayAdapter 로 단순문자열 리스트 또는 복합 아이템 리스틀 구성하고 클릭 이벤트를 처리 할 수 있다.
- ViewHolder 패턴을 적용해 getView 의 findViewByld 호출을 최소한 한다.

7.1 Adapter

7.1.1 Adapter 역할

Adapter는 배열, 파일, 데이터베이스에 저장된 데이터를 읽어서 화면에 표시할 때 유용한 뷰입니다.

이해를 돕기 위해 예를 들어 설명하면, 일반적으로 고가의 전자 장비에는 adapter 라는 전원 장치가 있다. 이러한 고가 장비는 전압(V)과 전류(A)가 중요한 요소인데, 전자 장비는 일정한 전압과 전류를 벗어나는 전압과 전류가 인가 되면 장비에 영향을 미치게 되고, 일반 가정집으로 들어오는 전압과 전류는 항상 일정하게 흐르지 않기 때문에 고가의 장비에 정해진 일정한 전압과 전류를 인가 받을 수 있도록 해주는 장치가 필요하다. 그것이 전원 adapter 장치의 역할이다.

즉, 안드로이드에서 Adapter 는 다시 말해 표시하고자 하는 항목별로 데이터를 원하는 형태로 변형하여 화면에 보여주는 역할을 하는 것이 adapter 이다.



그림 7-1 전압전류 변환장치 Adapter

표 7-1 전원 어댑터와 안드로이드 어댑터의 상관 관계

전원 어댑터	안드로이드 어댑터
전기를 기기에 맞게 변환	데이터를 화면에 맞게 변환
전압·전류 조절	View 형태로 가공
기기와 콘센트(전원)를 연결	데이터와 UI를 연결

위에서 언급한 안드로이드 Adapter 역할을 정리하면 아래와 같습니다.

- 데이터 소스 관리 :
  배열, 데이터베이스, 서버 API 등에서 데이터를 로딩
- 데이터를 View에 맞게 변환 :
  XML 레이아웃에 데이터를 바인딩하여 화면에 표시
- UI구성 최적화 :
  스크롤시 재활용(View Recycling)으로 메모리 절감
- 사용자 상호작용 원리 :
  리스트 클릭, 롱(Long) 클릭 등 이벤트 처리 가능

7.1.2 Adapter 동작 원리

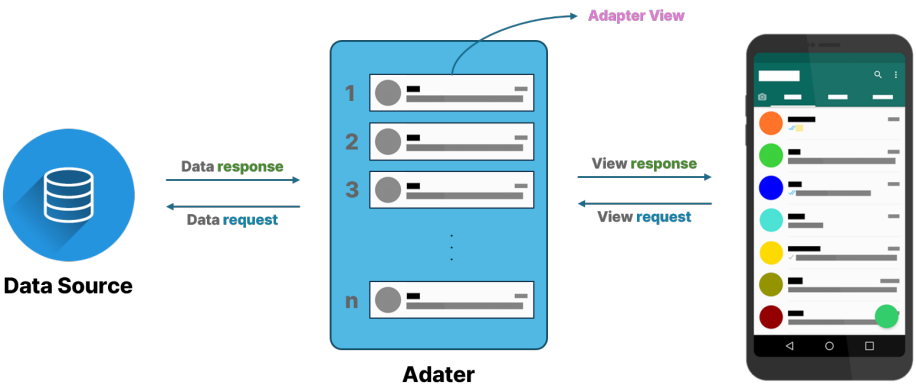


그림 7-2 안드로이드 Adapter 동작 흐름

데이터 소스는 데이터베이스, 배열, 네트워크 API 등 실제 정보를 가지고 있는 저장소를 의미한다. 어댑터는 이 데이터 소스에 데이터 요청을 보내고, 데이터 소스로부터 필요한 정보를 응답받는다. 이렇게 수신한 데이터는 그대로 화면에 표시할 수 없기 때문에, 어댑터는 이를 사용자 인터페이스에 적합한 형태로 가공한다.

가공된 데이터는 어댑터 뷰(Adapter View)로 전달된다. 어댑터 뷰(Adapter View) 는 ListView, RecyclerView, GridView와 같이 화면에 여러 개의 항목을 나열해 보여주는 UI 구성 요소이며, 이 뷰는 필요할 때 어댑터에 화면 요소(View)를 요청하며, 어댑터로부터 응답받은 뷰를 사용해 실제 화면에 그리게 됩니다.

최종적으로 사용자의 스마트폰 화면에는 어댑터 뷰를 통해 변환된 데이터가 리스트 형태로 표시된다. 이 과정에서 스크롤 동작 시 불필요한 뷰 생성을 최소화하기 위해 재활용(View Recycling) 기법이 사용되어 성능을 향상시키고, 데이터 변경 사항이 발생하면 어댑터가 즉시 뷰를 갱신하여 최신 정보를 반영하게 됩니다.

위 동작 내용을 동작별로 정리하면 아래와 같습니다.

- 데이터 소스 : 배열, JSON, DB, API 등
- 어댑터 객체생성 : 데이터와 레이아웃 연결
- View에 어댑터 연결 : 각 item에 대해 view와 adapter 연결
- getView() 메서드 호출 : 각 아이템을 화면에 표시
- 재활용(View Recycling) : 스크롤(scroll) 시 View를 재사용하여 성능 최적화

7.1.3 Adapter 종류

안드로이드는 여러 형태의 Adapter가 준비되어 있는데, 이는 사용자가 데이터 원본 종류에 따라 적합한 Adapter를 선택하여 상황에 맞게 활용하면 됩니다.

아래는 데이터 종류별 활용 가능한 Adapter를 정리한 내용입니다.

표 7-2 데이터별 Adapter

Adapter	설명	예시
ArrayAdapter	배열(Array) 또는 리스트(List)의 데이터를 간단한 표시	ArrayAdapter〈String〉
CursorAdapter	SQLite DB의 Cursor 데이터를 표시	채팅 기록, 연락처 목록
BaseAdapter	개발자가 직접 구현하는 맞춤형 어댑터	복잡한 UI 커스텀마이징
RecyclerView.Adapter	대량 데이터 최적화 표시	피드, 채팅, 뉴스목록

(1) ArrayAdapter

가장 기본적인 형태의 어댑터인 ArrayAdapter는 배열(Array)이나 리스트(List)와 같이 미리 정의된 간단한 데이터 집합을 화면에 손쉽게 표시할 수 있도록 도와주는 클래스입니다. 이 어댑터는 주로 안드로이드에서 자주 사용되는 ListView나 Spinner와 같은 위젯과 함께 활용되며, 데이터와 UI 간의 다리 역할을 수행합니다. 즉, 개발자가 별도의 복잡한 로직을 구현하지 않아도,

문자열이나 숫자와 같은 단순한 데이터를 손쉽게 화면에 나열할 수 있게 해줍니다. 예를 들어, 문자열 배열에 색상 이름이나 과일 이름과 같은 항목들을 저장해 두고 이를 ArrayAdapter와 연결하면, 사용자가 ListView를 통해 해당 데이터들을 스크롤하며 확인할 수 있습니다.

또한 ArrayAdapter는 안드로이드에서 제공하는 기본 레이아웃(android.R.layout.simple _list_item_1 등)을 그대로 활용할 수도 있고, 필요에 따라 개발자가 직접 정의한 레이아웃을 적용하여 커스터마이징할 수도 있습니다. 이러한 특성 덕분에 ArrayAdapter는 초보자에게는 데이터와 화면을 연결하는 기본적인 원리를 이해하는 데 좋은 출발점이 되고, 실무에서도 빠르게 간단한 목록형 UI를 구현하는 데 널리 사용됩니다.

따라서, ArrayAdapter는 가장 직관적이고 사용하기 쉬운 어댑터로, 안드로이드 앱 개발에서 데이터 바인딩의 기본을 익히는 데 매우 중요한 도구라고 할 수 있습니다.

아래는 ArrayAdapter 사용 예시입니다.

```
String[] fruits = {"사과", "바나나", "체리"};

ArrayAdapter<String> adapter
    = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, fruits);

ListView listView = findViewById(R.id.listView);
listView.setAdapter(adapter);
```



코드 설명

- String[] fruits = {“사과”, “바나나”, “체리”};
  → 데이터 소스를 준비하는 부분이며, String 타입 배열에 3개의 문자열 데이터 저장
- ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, fruits);
  → fruits 배열의 각 문자열을 simple_list_item_1 레이아웃을 사용해 하나씩 표시
- listView.setAdapter(adapter);
  → 어댑터를 ListView에 연결, ListView는 adapter로부터 데이터를 받아 화면에 표시
  → ListView는 스크롤이나 화면 업데이트가 필요할 때 어댑터의 getView()를 호출하여 각 항목을 렌더링 합니다.

(2) CursorAdapter

CursorAdapter는 안드로이드에서 데이터베이스와 UI를 연결하기 위해 자주 사용되는 어댑터 중 하나로, 특히 SQLite 데이터베이스에서 가져온 Cursor 객체를 화면에 표시할 때 활용됩니다. Cursor는 데이터베이스 쿼리 결과를 가리키는 일종의 포인터 역할을 하며, 이 Cursor의 데이터를 하나하나 꺼내어 리스트 형태의 UI에 표시하고자 할 때 CursorAdapter가 효과적으로 사용됩니다.

예를 들어, 연락처 목록이나 메시지 기록과 같이 사용자가 저장한 대규모 데이터를 데이터베이스에서 불러와야 하는 경우, 단순 배열을 사용하는 ArrayAdapter로는 한계가 있습니다. 이때 CursorAdapter를 활용하면 데이터베이스에 직접 연결된 Cursor로부터 데이터를 가져와 효율적으로 관리할 수 있으며, 새로운 데이터가 추가되거나 삭제될 때도 어댑터가 Cursor를 갱신하여 UI에 즉시 반영할 수 있습니다.

CursorAdapter의 가장 큰 장점은 데이터베이스와 UI 간의 직접적인 연동입니다. 개발자는 Cursor로부터 원하는 칼럼 값을 추출하여 리스트 아이템의 각 View에 바인딩하기만 하면 되므로, 데이터베이스의 구조와 UI의 구성이 유기적으로 연결됩니다. 또한 커스텀 레이아웃을 지정하여 단순한 텍스트 나열뿐만 아니라, 프로필 이미지, 이름, 부가 정보 등 다양한 UI 요소를 동시에 표시할 수도 있습니다.

따라서, CursorAdapter는 데이터베이스 기반의 앱, 예를 들어 연락처 앱, 메시지 앱, 메모 앱과 같이 지속적으로 DB와 상호작용하며 데이터를 표시하는 앱 개발에서 매우 적합하고 강력한 도구라고 할 수 있습니다.

아래는 CursorAdapter 사용 예시입니다.

```
Cursor cursor = db.rawQuery("SELECT name FROM contacts", null);

CursorAdapter adapter = new SimpleCursorAdapter(
    this,
    android.R.layout.simple_list_item_1,
    cursor,
    new String[]{"name"},
    new int[]{android.R.id.text1},
    0
);

ListView listView = findViewById(R.id.listView);
listView.setAdapter(adapter);
```

코드 설명

- Cursor cursor = db.rawQuery("SELECT name FROM contacts", null);
  → db.rawQuery()를 사용해 SQLite 데이터베이스에서 "contracts" 테이블의 "name" 컬럼 값을 로드하여 Cursor 객체 저장
- CursorAdapter adapter = new SimpleCursorAdapter(.....
  → simpleCursorAdapter를 이용해 cursor데이터를 simple_list_item_1 레이아웃의 text1 뷰 (android.R.id.text1)에 매핑
- listView.setAdapter(adapter);
  → 어댑터를 ListView에 연결, ListView는 adapter로부터 데이터를 받아 화면에 표시
  → ListView는 스크롤이나 화면 업데이트가 필요할 때 어댑터의 getView()를 호출하여 각 항목을 렌더링 합니다.

(3) BaseAdapter

BaseAdapter는 안드로이드에서 가장 기본적이면서도 범용적으로 활용할 수 있는 어댑터 클래스로, 개발자가 직접 레이아웃을 정의하고 데이터 바인딩 방식을 구현할 수 있도록 설계되어 있습니다. 다른 ArrayAdapter나 CursorAdapter처럼 특정 데이터 구조에 최적화된 어댑터와 달리, BaseAdapter는 제약이 적어 다양한 형태의 데이터와 복잡한 UI 구조를 자유롭게 처리할 수 있다는 장점이 있습니다.

즉, 단순히 텍스트만 나열하는 리스트가 아니라, 이미지, 버튼, 체크박스, 프로그레스바 등 여러 개의 뷰 요소를 하나의 아이템에 동시에 배치하고 싶은 경우, BaseAdapter를 활용하면 개발자가 원하는 형태로 레이아웃을 구성하고 데이터와 뷰를 원하는 방식으로 매핑할 수 있습니다. 또한, getView() 메서드를 직접 구현함으로써 각 아이템의 UI를 세밀하게 제어할 수 있으며, 데이터에 따라 동적으로 레이아웃을 변경하는 것도 가능합니다.

이러한 특징 덕분에 BaseAdapter는 단순한 리스트뷰뿐 아니라, 그리드뷰(GridView), 갤러리(Gallery), 커스텀 리스트뷰 등 다양한 UI 컴포넌트와 함께 사용되며, 특히 앱의 개성을 살린 맞춤형 UI를 구현할 때 널리 활용됩니다. 따라서, BaseAdapter는 단순 데이터 출력에서 벗어나 복합적인 사용자 경험을 제공하고자 할 때 매우 유용한 도구라고 할 수 있습니다.

아래는 BaseAdapter 사용 예시입니다.

```
public class MyAdapter extends BaseAdapter {
    private Context context;
    private String[] data;

    public MyAdapter(Context context, String[] data) {
        this.context = context;
        this.data = data;
    }

    @Override
    public int getCount() { return data.length; }

    @Override
    public Object getItem(int position) { return data[position]; }

    @Override
    public long getItemId(int position) { return position; }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = LayoutInflater.from(context)
                .inflate(android.R.layout.simple_list_item_1, parent, false);
        }
        TextView textView = convertView.findViewById(android.R.id.text1);
        textView.setText(data[position]);
        return convertView;
    }
}
```



코드 설명

- public class MyAdapter extends BaseAdapter {
  private Context context;
  private String[] data;
  → BaseAdapter를 상속받아 커스텀 어댑터 클래스(MyAdapter)를 구현.
- @Override
  public int getCount() { return data.length; }
  → 아이템 개수 반환.

- @Override
  public Object getItem(int position) { return data[position]; }
  → 특정 위치(position)에 해당하는 데이터 반환
- @Override

```
public View getView(int position, View convertView, ViewGroup parent) {
      if (convertView == null) {
          convertView = LayoutInflater.from(context)
              .inflate(android.R.layout.simple_list_item_1, parent, false);
      }
      TextView textView = convertView.findViewById(android.R.id.text1);
      textView.setText(data[position]);
      return convertView;
}
```

→ getView()메서드: 각 항목을 실제 View로 만들어 반환하는 핵심 부분
→ convertView 가 null(기존 뷰가 존재하지 않을 경우) 이면 레이아웃(XML)을 인플레이트하여 새로 생성
→ return convertview : 완성된 View를 반환하여 화면에 표시

(4) RecyclerView.Adapter

RecyclerView.Adapter는 안드로이드에서 대량의 데이터를 효율적이고 유연하게 화면에 표시하기 위해 설계된 어댑터 클래스로, 기존 ListView나 GridView의 단점을 보완하여 성능과 확장성을 크게 향상시킨 것이 특징입니다. 특히, ViewHolder 패턴과 함께 사용하도록 강제되어 있어, 불필요한 findViewById() 호출을 줄이고, 스크롤 시에도 뷰를 재활용함으로써 메모리 사용을 최소화하고 실행 속도를 최적화할 수 있습니다.

RecyclerView.Adapter는 데이터와 화면 요소를 연결하는 핵심 역할을 담당하며, 개발자가 onCreateViewHolder(), onBindViewHolder(), getItemCount()와 같은 필수 메서드를 구현하여 각 아이템의 레이아웃을 생성하고, 데이터를 뷰에 바인딩하며, 전체 데이터의 크기를 정의하게 됩니다. 이를 통해 수천 개 이상의 데이터도 성능 저하 없이 부드럽게 표시할 수 있으며, 복잡한 UI 요소를 가진 리스트나 그리드 구조도 효율적으로 관리할 수 있습니다.

또한, RecyclerView.Adapter는 단순히 리스트나 그리드 형식에 국한되지 않고, LayoutManager를 교체함으로써 세로 스크롤, 가로 스크롤, 격자(Grid), 계단식(Staggered) 등 다양한 형태의 배치

를 지원합니다. 이 덕분에 개발자는 데이터의 성격과 앱의 목적에 맞추어 UI를 자유롭게 구성할 수 있으며, 항목 추가 · 삭제 애니메이션, 드래그 앤 드롭, 스와이프 삭제와 같은 상호작용도 손쉽게 구현할 수 있습니다.

즉, RecyclerView.Adapter는 단순한 데이터 나열 도구를 넘어, 대규모 데이터 처리, 사용자 경험 최적화, 다양한 UI 패턴 구현을 가능하게 하는 강력한 어댑터라 할 수 있습니다.

아래는 RecyclerView.Adapter 사용예시입니다.

```
public class MyRecyclerAdapter extends RecyclerView.Adapter<MyRecyclerAdapter.ViewHolder>
{
    private String[] data;

    public static class ViewHolder extends RecyclerView.ViewHolder {
        TextView textView;
        public ViewHolder(View view) {
            super(view);
            textView = view.findViewById(android.R.id.text1);
        }
    }

    public MyRecyclerAdapter(String[] data) { this.data = data; }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
            .inflate(android.R.layout.simple_list_item_1, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        holder.textView.setText(data[position]);
    }

    @Override
    public int getItemCount() { return data.length; }
}
```

코드 설명

- public class MyRecyclerAdapter extends RecyclerView.Adapter<MyRecyclerAdapter.ViewHolder>

  → RecyclerView.Adapter를 상속받아 구현하며, 성능 최적화를 위해 ViewHolder 패턴을 사용하여 커스텀 어댑터를 구현

- ```
  public static class ViewHolder extends RecyclerView.ViewHolder {
      TextView textView;
      public ViewHolder(View view) {
          super(view);
          textView = view.findViewById(android.R.id.text1);
      }
  }
  ```

  → 각 아이템(View)를 보관하는 역할을 하며, 레이아웃 안에서 사용할 뷰 컴포넌트를 참조합니다.

- ```
  @Override
  public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
      View view = LayoutInflater.from(parent.getContext())
          .inflate(android.R.layout.simple_list_item_1, parent, false);
      return new ViewHolder(view);
  }
  ```

  → 새로운 ViewHolder 객체를 생성할 때 호출됩니다.

  → 레이아웃 인플레이터를 사용하여 simple_list_item_1 이라는 안드로이드 기본 단일 텍스트 항목 레이아웃을 객체로 변환 후 ViewHolder 객체를 생성하고 반환합니다.

- ```
  @Override
  public void onBindViewHolder(ViewHolder holder, int position) {
      holder.textView.setText(data[position]);
  }
  ```

  → 특정 위치의 데이터를 ViewHolder에 연결하는 역할을 합니다.

- ```
  @Override
  public int getItemCount() { return data.length; }
  ```

  → RecyclerView 가 전체 항목 개수를 알 수 있도록 데이터 배열의 길이를 반환합니다.

7.2 Adapter 실습

해당 부분에서는 기본적인 리스트 뷰를 활용한 Adapter와 사용자 정의 뷰를 활용한 Adapter를 실습해 보기로 합니다.

7.2.1 리스트(List) 뷰

가장 기본적인 리스트 뷰 실습을 위해 아래의 설정에 따라 프로젝트를 생성하세요.

- Project Name : listViewExam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) 레이아웃(XML) 디자인

레이아웃 디자인을 위해 [Code] 모드로 변경 후 아래와 같이 입력하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/lv_items"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

(2) 코드 작성

MainActivity.java

```
package com.example.listviewexam;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      // 리스트뷰에 표시할 문자열 데이터 배열
      String[] item1 = {"black", "yello", "blue", "orange", "gray", "red", "green"};

      // 레이아웃XML에 정의된ListView 가져오기
      ListView lv = (ListView) findViewById(R.id.lv_items);

      // ArrayAdapter 생성
      // → (context, 각 항목의 레이아웃, 데이터 배열)
      // → simple_list_item_1 : 안드로이드 기본 제공 단일 텍스트뷰 레이아웃
      ArrayAdapter<String> adapter =
            new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, item1);

      // 리스트 항목 클릭 이벤트 리스너 등록
      lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        // 사용자가 클릭한 아이템 값 가져오기
            String item = (String) parent.getItemAtPosition(position);

            // 토스트 메시지로 선택된 아이템 출력
            Toast.makeText(getApplicationContext(), "Selected Item :" + item,
                                                    Toast.LENGTH_LONG).show();
        }
      });

      // ListView에 어댑터 연결→ 화면에 데이터 표시 시작
      lv.setAdapter(adapter);
   }
}
```

위의 코드를 Adapter 동작 과정을 기반으로 설명하면 아래와 같습니다.



코드 설명

① 데이터 소스 준비 : 배열

- String[] item1 = {"black", "yellow", "blue", "orange", "gray", "red", "green"};

② 어댑터 객체 생성

- ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout. simple_list_item_1 , item1);
  - → ArrayAdapter<String>: 배열/리스트 같은 단순 컬렉션을 매핑하는 기본 어댑터이며, 데이터 타입은 문자열(String)로 지정.
  - → 배열 item1 데이터를 simple_list_item_1 레이아웃 형태로 표시

③ 어댑터 연결

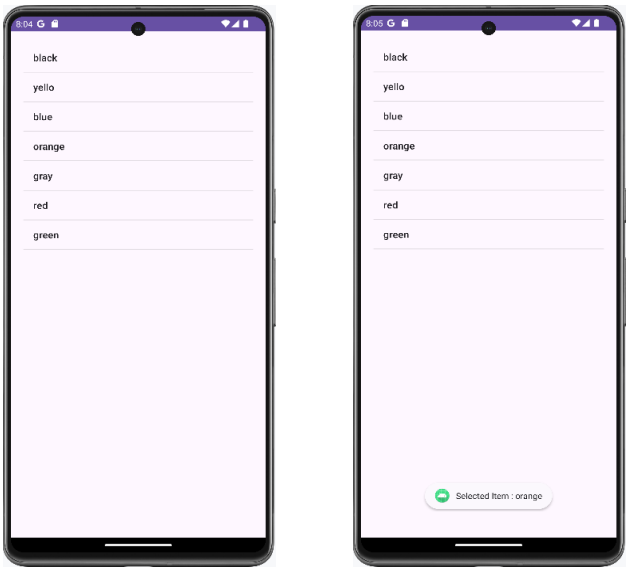
- lv.setAdapter(adapter);
  - → lv는 id 가 lv_items 라는 ListView 컴포넌트를 지칭(AdapterView).
  - → lv에 ArrayAdapter 클래스 객체인 adapter를 연결(setAdapter).

④ 표시된 항목 터치

- @Override

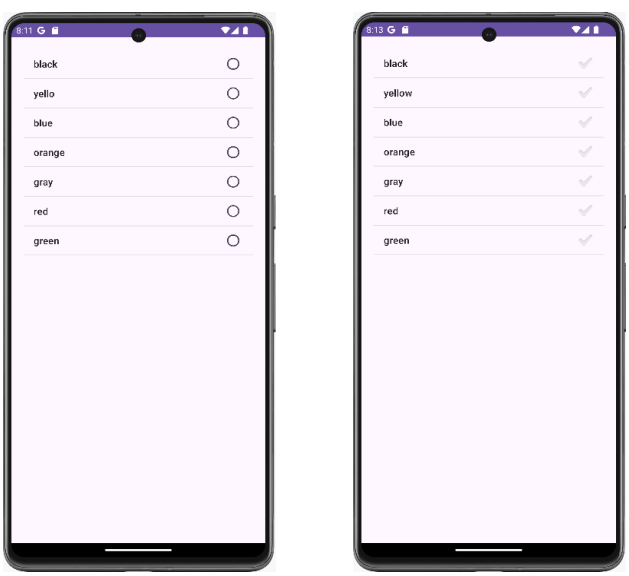
```
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        String item = (String) parent.getItemAtPosition(position);
        Toast.makeText(getApplicationContext(), "Selected Item : " + item,
                                                    Toast.LENGTH_LONG).show();
}
```

- → 파라메터 parent: 클릭이 발생한 AdapterView (lv: ListView)
- → 파라메터 view : 클릭된 행(Row)의 View
- → 파라메터 position : 클릭된 항목의 인덱스(위치) 값
- → 파라메터 id : 클릭된 항목의 어댑터 자체로 제공하는 아이디(positon과 동일)
- → parent.getItemAtPosition(position) : 클릭된 항목을 위치값 기반하여 데이터를 가져옴.



리스트 항목에서 특정 항목을 터치하여 결과를 확인하세요.

위 코드에서 android.R.layout.simple_list_item_1을 android.R.simple_list_item_single_choi ce, android.R.layout.simple_list_item_checked 로 변경해 보면 아래와 같이 리스트 항목 표시가 다르게 나타나는 것을 알 수 있을 것입니다.



simple_list_item_single_choice simple_list_item_checked

7.2.2 리스트(List) 뷰 응용

7.2.1 에서 실습한 내용을 바탕으로 리스트 뷰를 약간 변형하려고 합니다.

해당 프로젝트에서는 사용자가 직접 데이터를 입력받아 리스트 항목을 추가하고, 추가된 항목을 수정하거나 삭제하는 기능을 갖도록 구성할 계획입니다.

실습을 위해 아래의 설정에 따라 프로젝트를 생성하세요.

- Project Name : listView2Exam
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) 레이아웃 디자인

레이아웃 디자인을 위해 [Code] 모드로 변경 후 아래와 같이 입력하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/btn_add"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="추가"
        android:textSize="18dp" />

    <Button
        android:id="@+id/btn_modify"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="100dp"
        android:text="수정"
        android:textSize="18dp" />
```

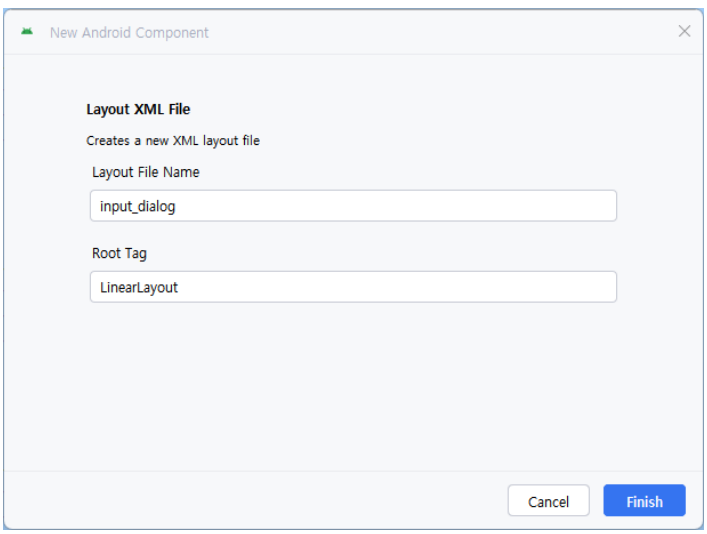
```
    <Button
        android:id="@+id/btn_delete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="200dp"
        android:text="삭제"
        android:textSize="18dp" />

    <ListView
        android:id="@+id/lv_items"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/btn_add" />
</RelativeLayout>
```

(2) 대화상자 레이아웃

사용자가 직접 데이터를 추가, 수정, 변경하기 위해 사용자 입력용 대화상자가 필요합니다. 사용자 입력용 대화상자의 디자인을 아래와 같이 생성하고, 코드를 입력하세요.

- app 〉 res 〉 layout 마우스 우클릭 후 new 〉 XML 〉 Layout XML File 선택
- Layout File Name : input_dialog



input_dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="20dp">

    <EditText
        android:id="@+id/et_in"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:layout_weight="1"
        android:ems="10"
        android:hint="여기에 클릭 후 입력하세요."
        android:inputType="text"
        android:textSize="15dp"
        tools:ignore="TextSizeCheck" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_weight="1"
        android:orientation="horizontal"
        android:padding="5dp">

        <Button
            android:id="@+id/btn_ok"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="10dp"
            android:layout_weight="1"
            android:text="확인" />

        <Button
            android:id="@+id/btn_cancle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="취소" />
    </LinearLayout>
</LinearLayout>
```

(3) 코드 작성

MainActivity.java

```
package com.example.listview2exam;

import android.app.Dialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    ArrayAdapter<String> adapter;
    ArrayList<String> items;  // 데이터 소스
    ListView lv;

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

        items = new ArrayList<String>();
        adapter = new ArrayAdapter<String>(this,
                                android.R.layout.simple_list_item_single_choice, items);

        lv = (ListView) findViewById(R.id.lv_items);
        lv.setAdapter(adapter);
        lv.setChoiceMode(ListView.CHOICE_MODE_SINGLE);

        // 버튼 객체에 클릭 리스너 추가
        findViewById(R.id.btn_add).setOnClickListener(this);
        findViewById(R.id.btn_modify).setOnClickListener(this);
        findViewById(R.id.btn_delete).setOnClickListener(this);
    }

    @Override
```

```
    public void onClick(View v) {
        int id = v.getId();

        if(id == R.id.btn_add) {
            data_add();
        } else if(id == R.id.btn_modify) {
            data_modify();
        } else {
            data_delete();
        }
    }

    //데이터 추가 메서드
    public void data_add() {
        Dialog custom = new Dialog(this);
        // "추가 버튼 클릭시 표시할 레이아웃( input_dialog.xml )
        custom.setContentView(R.layout.input_dialog);
        custom.show();

        // 다이얼로그 객체custom 의 버튼 정의
        Button btn_ok = (Button) custom.findViewById(R.id.btn_ok);
        Button btn_cancle = (Button) custom.findViewById(R.id.btn_cancle);

        //  다이얼로그 객체custom 의 사용자 입력 박스(EditText) 정의
        EditText et_in = (EditText) custom.findViewById(R.id.et_in);

        // 확인 버튼 클릭시
        btn_ok.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(et_in.getText().toString().trim().length() > 0) {
                    // 사용자가 입력한 값(EditText)을 데이터소스에 추가
                    items.add(et_in.getText().toString());
                    // 데이터소스에 변동이 발생-> 어댑터에 알림
                    adapter.notifyDataSetChanged();
                    custom.dismiss(); //다이얼로그 박스 닫기
                } else { // 사용자 입력 값이 없을 경우
                    custom.dismiss();
                }
            }
        });

        //취소 버튼 클릭시
        btn_cancle.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                custom.dismiss();
            }
        });
    }

    // 데이터 수정 메서드
    public void data_modify() {
        final Dialog custom = new Dialog(this);

        custom.setContentView(R.layout.input_dialog);

        Button btn_ok = (Button) custom.findViewById(R.id.btn_ok);
        Button btn_cancle = (Button) custom.findViewById(R.id.btn_cancle);

        final EditText et_in = (EditText) custom.findViewById(R.id.et_in);
        // 선택한 항목에 대한 값을 불러와EditText에 표시
        et_in.setText(items.get(lv.getCheckedItemPosition()));
        custom.show(); // 다이얼로그 박스 표시

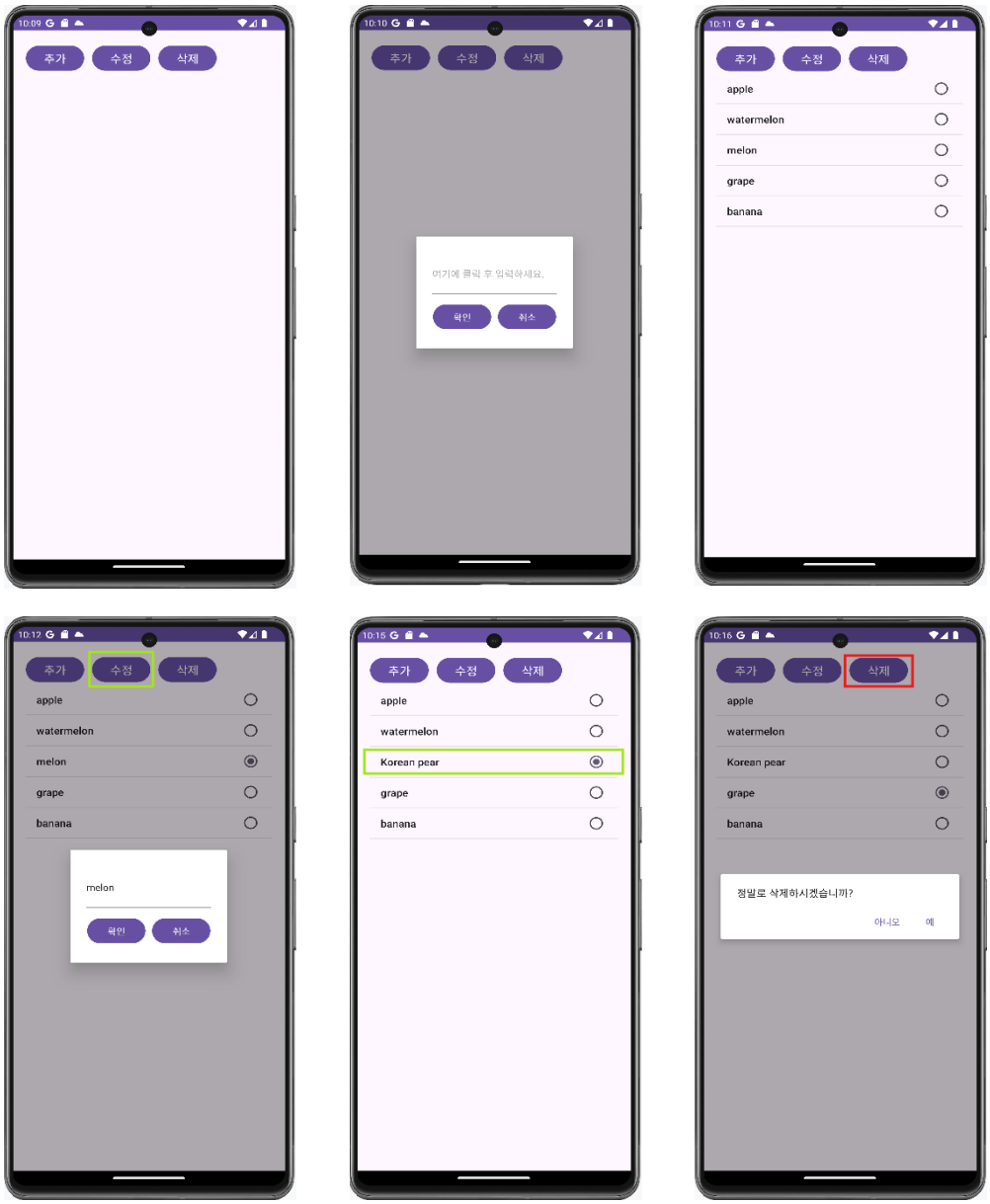
        btn_ok.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int count, checked;
                count = adapter.getCount(); // 현재 데이터 개수 파악

                if(et_in.getText().toString().trim().length() > 0) {
                    if(count > 0) {
                        //선택한 항목 위치값
                        checked = lv.getCheckedItemPosition();
                        // 선택한 항목 위치값이 정상일 때(만약의 오류 대비)
                        if(checked > -1 && checked < count) {
                    // 데이터 소스를 변경/ 선택 위치(checked)에 수정된 값으로 변경
                            items.set(checked, et_in.getText().toString());
                            adapter.notifyDataSetChanged();
                        }
                    }
                    custom.dismiss();
                } else {
                    custom.dismiss();
                }
            }
        });
        btn_cancle.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                custom.dismiss();
```

```
            }
        });
    }

    // 삭제 버튼 클릭시
    // 추가, 수정 버튼 클릭과 다르게AlertDialog 활용
    public void data_delete() {
        AlertDialog.Builder dialog = new AlertDialog.Builder(this);
        dialog.setMessage("정말로 삭제하시겠습니까?");
        dialog.setPositiveButton("예", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                int count, checked;
                count = adapter.getCount();

                if(count > 0) {
                    checked = lv.getCheckedItemPosition();
                    if(checked > -1 && checked < count) {
                        //데이터 소스에서 선택항목 삭제
                        items.remove(checked);
                        lv.clearChoices(); // ListView 다시 그리기
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        });
        dialog.setNegativeButton("아니오", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                    ;
            }
        });
        AlertDialog dlg = dialog.create();
        dlg.show();
    }
}
```



7.2.2 커스텀 Adapter 실습 (BaseAdapter 기반)

이전 섹션까지는 안드로이드가 기본 제공하는 표준 어댑터를 사용해 리스트를 구성했습니다. 하지만 기본 어댑터만으로는 항목별 개별 디자인을 자유롭게 바꾸기 어렵습니다. 이번 섹션에서는 BaseAdapter를 상속해 필요한 메서드를 오버라이드(Override)하고, XML 레이아웃과 클

래스 기반의 복합 데이터 모델을 결합하여 사용자가 원하는 형태로 항목을 표현하는 커스텀 어댑터를 구현합니다.

앞서 한 항목에 텍스트 한 줄만 표시했다면, 이제는 영화 예매 앱처럼 항목 하나에 썸네일, 감독, 출연진, 상영 기간 등 여러 정보를 동시에 배치할 수 있습니다. 즉, 리스트의 각 아이템을 별도 레이아웃으로 설계하고 데이터 모델에 매핑함으로써, 실제 서비스 수준의 풍부한 표현력을 갖춘 리스트를 만들어 보겠습니다.

이번 프로젝트 실습에서 이전 프로젝트와 다르게 새로 추가되는 작업은 아래와 같습니다.

- 데이터 모델(클래스) 설계
  - 연관 데이터들을 하나의 객체로 묶음으로 관리 용이하고, 명확해 짐.
- Adapter View 레이아웃 제작
  - 객체로 묶여진 데이터를 사용자가 원하는 레이아웃으로 구성
  - 항목별 일관성 있는 디자인 가능
- 커스텀 Adapter 클래스 구현 (BaseAdapter 상속)

이제 실제 구현을 진행하도록 하겠습니다. 아래와 같이 설정하고 프로젝트를 생성하세요.

- Project Name : customList
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

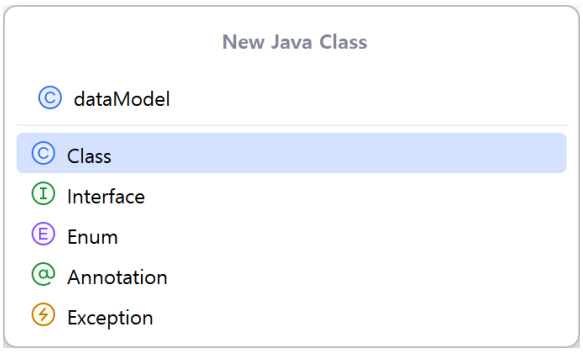
단, 해당 프로젝트에서는 외부에서 데이터를 불러오는 형태는 아니고, 코드 내부에서 데이터를 설정하고 이 데이터를 리스트로 표현하는 프로젝트로 구성하도록 할 것입니다. 네트워크를 통해 외부데이터를 불러오는 실습은 다음 챕터 8장에서 다뤄보도록 하겠습니다.

(1) 데이터 모델 (클래스)

복합 데이터들을 하나의 묶음으로 관리할 수 있도록 클래스를 구현하는 단계입니다. 클래스로 구현을 하지 않고 복합 데이터를 개별로 관리한다면, 데이터를 개별로 불러와 처리해야 하므로 복잡해 집니다. 만약 하나의 항목에 표시할 데이터가 5개 정도라고 가정한다면, 5 개 데이터를 인자로 받아들여야 하는 번거로움이 발생하여 코드가 복잡해 집니다. 이 5개 데이터를 클래스로 묶은 객체를 한 번에 전달한다면 편리해 지는 것은 당연한 논리입니다.

아래 안내하는 폴더에 파일을 생성하고 코드를 작성하세요.

- app 〉 java 〉 com.example.customlist를 선택하고, 마우스 우클릭
- new 〉 Java Class 선택
- 클래스명 "dataModel" 로 입력



dataModel.java

```
package com.example.customlist;

import android.graphics.drawable.Drawable;

/**
 * 리스트의 한 항목을 표현하는 데이터 모델.
*/
public class dataModel {
   private Drawable icon;
   private String name;  /** 표시용 이름(예: 연락처 이름) */
   private String phone; /** 전화번호(형식 제약 없음: 010-XXXX-XXXX 등) */


   // class 생성자
   public dataModel(long id, Drawable icon, String name, String phone) {
      this.id = id;
      this.icon = icon;
      this.name = name;
      this.phone = phone;
   }

   /** icon 을 반환합니다.  접근자 */
```

```
    public Drawable getIcon() {  return icon;    }

    /** icon 을 설정/변경합니다.  설정자  */
    public void setIcon(Drawable icon) {   this.icon = icon;   }

    /** name을 반환합니다.  접근자 */
    public String getName() {   return name;    }

    /** name 을 설정/변경합니다.  설정자  */
    public void setName(String name) {   this.name = name;    }

    /** phone 을 반환합니다.  접근자 */
    public String getPhone() {   return phone;    }

    /** phone 을 설정/변경합니다.  설정자  */
    public void setPhone(String phone) {   this.phone = phone;   }
}
```



코드 설명

① import android.graphics.drawable.Drawable;

- 안드로이드의 이미지 표현 타입인 Drawable 라이브러리 임포트

② private Drawable icon;

- 그림 이미지 파일을 담을 수 있는 변수 선언

③ public dataModel(Drawable icon, String name, String phone) {

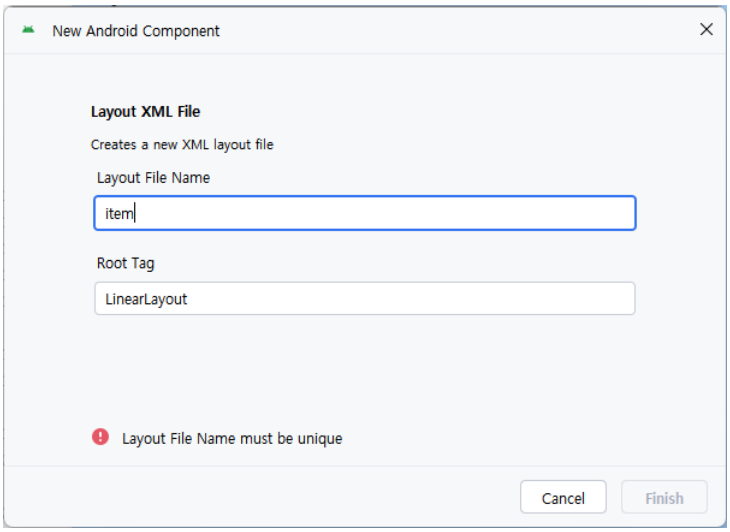
- 생성자를 뜻하며, 객체 생성시 파라메터(icon, name, phone)를 통해 값을 받음.
  → this.icon = icon; 파라메터 icon 변수로 받은 값을 클래스 변수(this.icon)에 전달
  → this.name = name; 파라메터 name 변수로 받은 값을 클래스 변수(this.name)에 전달
  → this.phone = phone; 파라메터 phone 변수로 받은 값을 클래스 변수(this.phone)에 전달

(2) Adapter View 레이아웃 설계

항목별 데이터 표시 레이아웃을 구성하는 단계입니다.

- app 〉 res 〉 layout를 선택하고, 마우스 우클릭
- new 〉 XML 〉 Layout XML File 클릭

- Layout File Name 항목에 “item” 입력



File 생성 후 아래와 같이 입력하세요.

item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="20dp">

    <ImageView
        android:id="@+id/iv_icon"
        android:layout_width="80dp"
        android:layout_height="80dp"
        app:srcCompat="@mipmap/ic_launcher" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:orientation="vertical"
        android:paddingLeft="30dp">

        <TextView
```

```
            android:id="@+id/tv_name"
            android:layout_width="match_parent"
            android:layout_height="40dp" />

        <TextView
            android:id="@+id/tv_phone"
            android:layout_width="match_parent"
            android:layout_height="40dp" />
    </LinearLayout>
</LinearLayout>
```

(3) 커스텀 Adapter 클래스 설계 (BaseAdapter 상속)

앞 단계에서 설정한 레이아웃에 데이터소스(원본)를 어떤 뷰에 표시할 것인지, 어떻게 표시할 것인지를 설계하는 단계입니다.

- app 〉 java 〉 com.example.customlist를 선택하고, 마우스 우클릭
- new 〉 Java Class 선택
- 클래스명 "itemApdater" 로 입력

File 생성 후 아래와 같이 입력하세요.

itemAdapter.java

```
package com.example.customlist;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;

public class itemAdapter extends BaseAdapter {

    /** 화면에 표시할 데이터 목록(아이템 단위). */
    private ArrayList<dataModel> mItems = new ArrayList<dataModel>();

    /**
```

```
 * ViewHolder: findViewById 호출을 캐싱하여 성능을 높이는 보조 클래스.
 * static으로 선언하면 외부 클래스에 대한 암묵적 참조를 막아 누수 위험을 더 낮출 수 있음(선택).
 */
    public class CustomViewHolder {
        ImageView iv_icon;   // 썸네일/프로필 아이콘
        TextView  txt_name;  // 이름
        TextView  txt_phone; // 전화번호
    }

 /**
 * 어댑터의 데이터에 한 건을 추가한다.
 * @param item 리스트에 추가할 dataModel
 */
    public void addItem(dataModel item) {
        mItems.add(item);
    }

 /** 아이템 총 개수 반환(ListView가 스크롤 범위를 계산할 때 사용). */
    @Override
    public int getCount() {
        return mItems.size();
    }

 /** position에 해당하는 아이템을 반환. (BaseAdapter 규약상 Object 타입) */
    @Override
    public Object getItem(int position) {
        return mItems.get(position);
    }

 /**
 * 아이템의 ID 반환. 현재는 position을 그대로 반환하므로 '안정 ID'가 아님.
 * 모델에 고유 id가 있다면 그 값을 반환하고 hasStableIds()를 true로 오버라이드하는 방식 권장.
 */
    @Override
    public long getItemId(int position) {
        return position;
    }

 /**
 * 핵심 메서드: position의 데이터를 재활용 뷰(convertView)에 바인딩하여 화면에 표시할 뷰를 반환.
 *
 * @param position    데이터 인덱스
 * @param convertView 재활용 가능한 기존 뷰(없으면 null)
 * @param parent      이 뷰가 속하게 될 상위 뷰(레이아웃 파라미터 계산에 사용)
 * @return 표시할(또는 재활용한) 아이템 뷰
 */
```

```
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        CustomViewHolder holder;

        if (convertView == null) {
//아직 재활용 뷰가 존재하지 않으므로 새로운 인플래이트(inflate) 생성
// 권장: parent를 루트로 전달하고attachToRoot=false로 해야 레이아웃 파라미터가 올바르게 적용됨

            convertView = LayoutInflater.from(parent.getContext())
                                        .inflate(R.layout.item, null, false);

            // ViewHolder 생성 및 하위 뷰 캐싱
            holder = new CustomViewHolder();
            holder.iv_icon  = (ImageView) convertView.findViewById(R.id.iv_icon);
            holder.txt_name = (TextView)  convertView.findViewById(R.id.tv_name);
            holder.txt_phone= (TextView)  convertView.findViewById(R.id.tv_phone);

            // 재활용 시 꺼내 쓰기 위해 태그로 보관
            convertView.setTag(holder);

        } else {
        // 재활용 가능한 뷰가 있으므로, 저장해둔ViewHolder를 꺼내 사용
            holder = (CustomViewHolder) convertView.getTag();
        }

        // position에 해당하는 데이터 모델 조회
        dataModel item = mItems.get(position);

        // 데이터→ 뷰 바인딩
        holder.iv_icon.setImageDrawable(item.getIcon());
        holder.txt_name.setText(item.getName());
        holder.txt_phone.setText(item.getPhone());

        // 완성된(또는 재활용한) 뷰 반환
        return convertView;
    }
}
```

코드 설명

① public class itemAdapter extends BaseAdapter {

→ BaseApdater를 상속해 itemAdapter 클래스 선언

② private ArrayList<dataModel> mItems = new ArrayList<dataModel>();

→ 한 행(row)에 필요한 정보를 담은 dataModel 객체 리스트 변수 mItems 선언

③ public class CustomViewHolder { .....

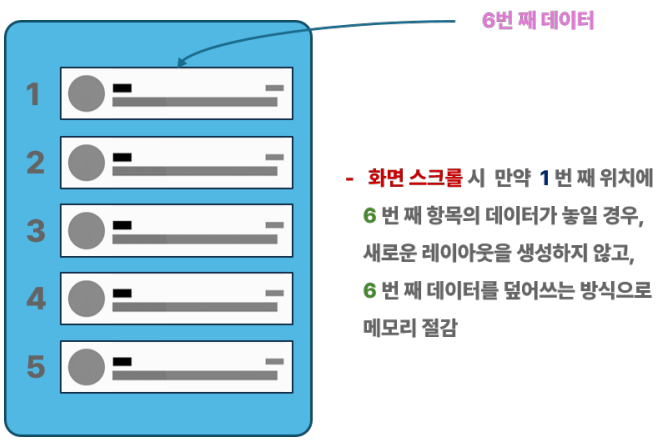
→ ViewHolder 패턴을 선언으로 GC 압박과 반복 탐색 비용을 줄여 스크롤 성능 향상

→ 하위 코드 중 findViewById로 찾아온 하위 뷰 참조를 보관해, 스크롤 중 재사용시 매번 찾지 않도록 함.

④ public View getView(int position, View convertView, ViewGroup parent) {

→ convertView: 스크롤할 때마다 새 뷰를 만들면 인플레이트 비용이 커서 끊김이 발생

→ ListView는 스크린 밖으로 나간 뷰를 convertView에 저장하며, 어댑터는 이를 채워서 재사용합니다.



(4) 메인 레이아웃 (activity_main.xml)

개별 항목 레이아웃(item.xml)을 표시하기 위한 메인 레이아웃 구성 단계입니다. 아래 안내에 따라 입력해 주세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/lv_items"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_margin="10dp" />
</LinearLayout>
```

(5) 코드 작성

코드를 작성하기 전에 코드 수행에 필요한 이미지 파일을 app 〉 res 〉 Drawable 폴더에 복사해 주세요. 이미지 파일은 다운로드 받은 폴더 내에 face01.png~face05.png 파일입니다. 이제 모든 구성이 완료되었습니다. 마지막으로 아래 코드를 작성하여 실행하여 결과를 확인해 보세요.

MainActivity.java

```
`package com.example.customlist;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private ListView lv_list;       // 메인 레이아웃의ListView (아이템 목록 표시)
    public itemAdapter adapter;     // 커스텀 어댑터(데이터→ item.xml 바인딩 담당)

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);

        // 레이아웃에서ListView 참조 획득
        lv_list = (ListView) findViewById(R.id.lv_items);

        // 커스텀 어댑터 생성(BaseAdapter 상속 구현)
        adapter = new itemAdapter();

        // ===== (중요) 어댑터에 표시할 데이터 주입=====
        // ContextCompat.getDrawable: 호환성 있는 방식으로Drawable 로드
        // dataModel: 아이콘/이름/전화번호를 한 덩어리(모델)로 보관
        adapter.addItem(new dataModel(
            ContextCompat.getDrawable(getApplicationContext(), R.drawable.face01),
                                                    "홍길동", "010-1234-1234"));

        adapter.addItem(new dataModel(
            ContextCompat.getDrawable(getApplicationContext(), R.drawable.face02),
                                                     "이순신", "010-1212-1212"));

        adapter.addItem(new dataModel(
            ContextCompat.getDrawable(getApplicationContext(), R.drawable.face03),
                                                     "아이유", "010-4534-1245"));

        adapter.addItem(new dataModel(
            ContextCompat.getDrawable(getApplicationContext(), R.drawable.face04),
                                                    "블랙핑크", "010-6134-6234"));

        adapter.addItem(new dataModel(
            ContextCompat.getDrawable(getApplicationContext(), R.drawable.face05),
                                                      "BTS", "010-1324-1434"));

        // (중요) ListView와 어댑터 연결→ 이제ListView가 어댑터를 통해 행을 그림
        lv_list.setAdapter(adapter);

        // (중요) 아이템 클릭 리스너 등록
        // 사용자가 특정 행을 탭하면 해당position의dataModel을 꺼내 토스트로 이름 표시
        lv_list.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                // position 기반으로 모델 획득
                dataModel item = (dataModel) adapter.getItem(position);
                Toast.makeText(getApplicationContext(), item.getName(), Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

코드 설명

① adapter = new itemAdapter();

```
adapter.addItem(new dataModel(
        ContextCompat.getDrawable(getApplicationContext(), R.drawable.face01),
                                                    "홍길동", "010-1234-1234"));
```

→ 커스텀 어댑터 클래스(itemAdapter) 인스턴스 변수 adapter 생성

→ 인스턴스 변수 adapter에 생성된 모델 클래스(dataModel: 이미지, 성명, 전화번호) 객체를 대입

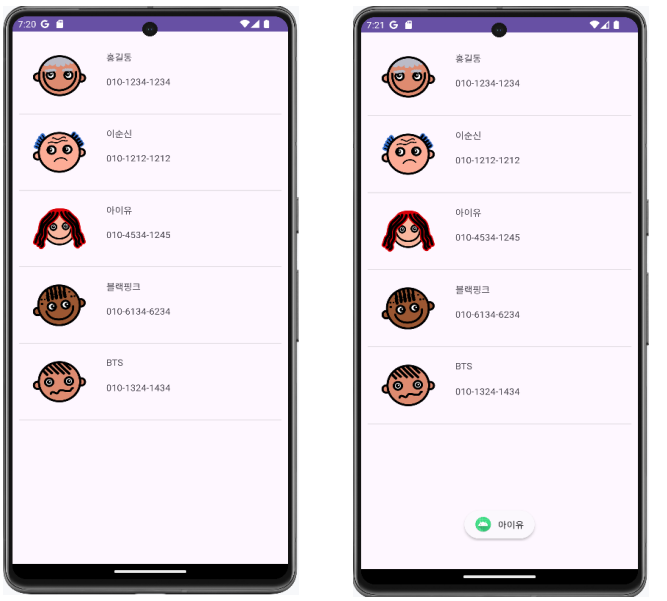
② lv_list.setAdapter(adapter);

→ ListView(lv_list)를 커스텀 어댑터의 인스턴스 변수인 adapter와 연결

③ lv_list.setOnItemClickListener(new AdapterView.OnItemClickListener() {

.....

→ 아이템 항목 클릭시 처리할 코드



여기까지 커스텀 Adapter를 설계하여 구현한 간단한 프로젝트 예제였습니다. 이번 예제는 여러분의 이해를 돕기 위해 최소한의 기능만을 포함했지만, 대화상자(Dialog), 날짜 · 시간 Picker와 같은 다양한 위젯이나 기능을 응용하여 어플리케이션을 제작해 본다면, 커스텀 Adapter에 대한 활용 능력과 이해도를 한층 더 높일 수 있을 것입니다.



연습문제

1. Adapter의 개념과 역할에 대해 설명하세요.

2. Adapter 동작 원리를 구체적으로 설명하시오. 데이터 소스에서 화면에 표시되기까지의 과정(데이터 소스 → 어댑터 → 어댑터 뷰)을 단계별로 기술하세요.

3. Adapter 종류(ArrayAdapter, CursorAdapter, BaseAdapter, RecyclerView .Adapter)의 특징과 차이를 설명하고, 각각 어떤 상황에서 사용하는 것이 적합한지 실제 예시를 들어 설명하세요.

4. BaseAdapter와 RecyclerView.Adapter의 차이점을 설명하세요. 특히 ViewHolder 패턴 적용 방식과 성능 최적화 측면에서의 차이를 중심으로 기술하시오.

5. BaseAdapter를 이용한 커스텀 Adapter는 왜 필요한지, 기본 Adapter와 비교하여 설명하시오.



제8장

앱 개발



학습목표

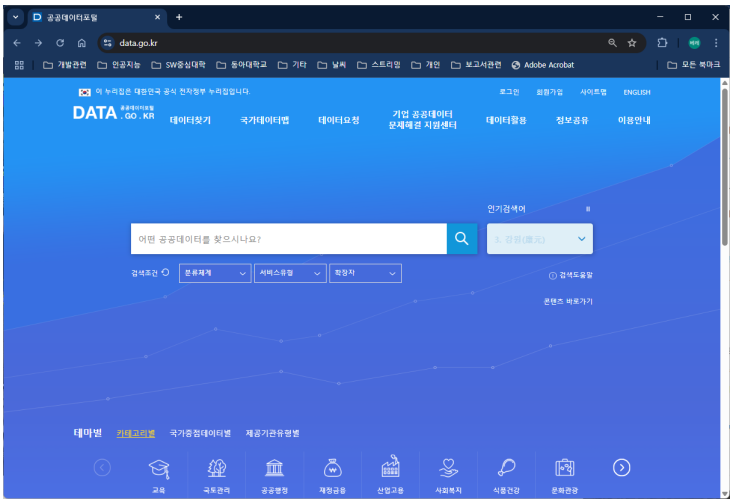
- 공공데이터(Open API) 구조와 사용 절차를 설명하고, 필요한 인증 요청 · 응답 과정을 직접 수행할 수 있다.
- JSON/XML 응답을 파싱 · 정제하여 앱에서 활용 가능한 도메인 모델로 매핑할 수 있다.
- 안드로이드에서 네트워킹(비동기 요청)과 리스트 UI(커스텀 어댑터)를 설계, 구현할 수 있다.
- 예외 · 네트워크 오류 · 빈 데이터 등을 처리하고, 로딩/에러/성공 상태를 표현할 수 있다.

8.1 공공 데이터 활용방법

8.1.1 공공데이터 사전 준비

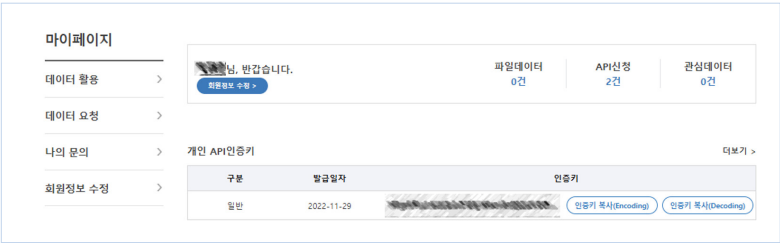
국내 공공데이터는 정부에서 운영하는 공공데이터포털(http://www.data.go.kr)을 통해 누구나 자유롭게 검색 · 활용할 수 있으며, 이를 기반으로 안드로이드 앱뿐만 아니라 웹 프로그램, 데이터 분석 서비스 등 다양한 형태의 응용 서비스 개발이 가능합니다.

여러분도 먼저 해당 사이트에 접속하여 회원 가입을 진행해 주시기 바랍니다. 회원 가입 절차는 일반적인 웹 서비스와 크게 다르지 않으므로, 본 교재에서는 별도의 상세 과정을 다루지 않겠습니다.



다만, 회원 가입을 마친 뒤 반드시 로그인하여 [마이페이지] 메뉴를 확인하는 것이 중요합니다. 이 메뉴에서는 사용자의 계정과 관련된 핵심 정보를 확인할 수 있으며, 그중에서도 앱 개발 과정에서 반드시 필요한 개인 API인증키를 확인할 수 있습니다. (단, 개인API인증키는 공공데이터를 최소 한 건 이상 활용 신청해야 발급 및 표시됨)

이 API 인증키는 공공데이터를 애플리케이션에서 호출할 때 반복적으로 사용되는 필수 정보이므로, 반드시 안전하게 보관하고 외부에 유출되지 않도록 각별히 주의해야 합니다. 특히 인증키가 유출되면 본인의 계정이 악용되거나 서비스 이용에 제한이 발생할 수 있으므로, 초기 로그인 이후 반드시 확인하는 방법과 안전한 관리 방법을 익혀 두는 것이 중요합니다.



즉, 공공데이터포털을 활용한 첫 단계는 단순한 회원 가입을 넘어, 로그인 후 마이페이지에서 API 인증키를 확인하고 이를 안전하게 관리할 수 있는 습관을 갖추는 것이라 할 수 있습니다. 이를 통해 이후 앱 개발 과정에서 필요한 공공데이터를 안정적이고 효율적으로 활용할 수 있습니다.

표 8-1 공공데이터 활용한 앱 개발을 위한 준비 단계

단계	주제	수행 내용
1단계	공공데이터 포털 접속	• 웹브라우저 주소창에 http://www.data.go.kr 를 입력 • 공공데이터포털 첫화면에서 회원가입 버튼 클릭
2단계	회원 가입	• 회원가입 진행 • [로그인] 버튼으로 계정 로그인
3단계	마이페이지 접속	• [마이페이지] 메뉴 클릭 • 사용자 계정 관련 정보와 서비스 이용내용 확인 가능
4단계	개인 API 인증키 확인	• [마이페이지] 화면에서 개인 API 인증키 발급여부 확인
5단계	API 인증키 보관 및 관리	• 발급받은 개인API 인증키는 안전 보관 필수

8.2 공공데이터 활용 앱 개발

앞선 학습 과정을 통해 우리는 공공데이터포털에 회원 가입을 완료하고, 로그인 후 개인 API 인증키를 발급받는 등 데이터를 활용하기 위한 기본적인 준비를 마쳤습니다. 이러한 과정을 거침으로써 이제는 실제로 공공데이터를 활용할 수 있는 기반이 마련된 것입니다. 이제 다음 단계로, 사용자가 필요로 하는 데이터를 직접 검색하고, 검색된 데이터를 어떤 방식으로 내려받거나 연동하여 활용할 수 있는지 구체적으로 살펴보겠습니다.

특히 공공데이터포털에서는 CSV, XLS, JSON과 같은 파일 다운로드 방식과 Open API를 통한 실시간 연동 방식을 모두 지원하기 때문에, 사용 목적에 따라 적절한 활용 방법을 선택하

는 것이 중요합니다. 이러한 두 가지 활용 방식을 중심으로, 데이터를 앱이나 웹 서비스 개발 과정에서 어떻게 적용할 수 있는지 간단한 예시를 통해 알아보도록 하겠습니다.

본 교재에서는 공공데이터를 활용한 안드로이드 앱 개발 학습을 위해 연습용으로 자주 사용되는 "농림축산식품부 농림축산검역본부_국가동물보호정보시스템 구조동물 조회 서비스"를 활용하고자 합니다. 이 API는 대기질에 대한 다양한 정보를 실시간으로 제공하기 때문에 학습자들이 실제 데이터를 기반으로 앱을 개발해 보는 데 매우 유용합니다.

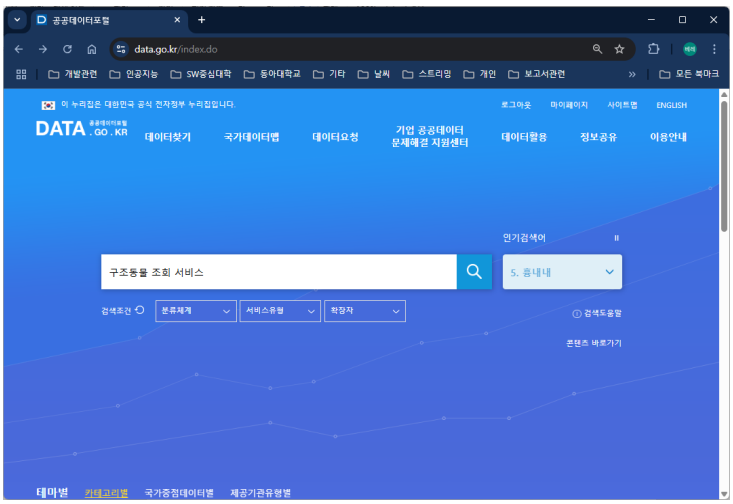
이제부터는 학습자 여러분이 직접 공공데이터 포털에서 해당 API를 발급받고, 이를 안드로이드 앱 개발 환경에서 연동하여 사용하는 과정을 단계별로 실습할 수 있도록 안내하겠습니다. 각 단계에서는 데이터 검색 및 다운로드, 인증키 발급, API 호출을 통한 데이터 수집, 그리고 수집된 데이터를 안드로이드 애플리케이션에 적용하는 순서를 차근차근 설명합니다.

따라서 아래 제시된 절차를 차례대로 따라가면서 실습을 진행하면, 단순한 예제 코드를 넘어 실제 공공데이터를 활용한 응용 프로그램을 제작하는 경험을 쌓을 수 있을 것입니다.

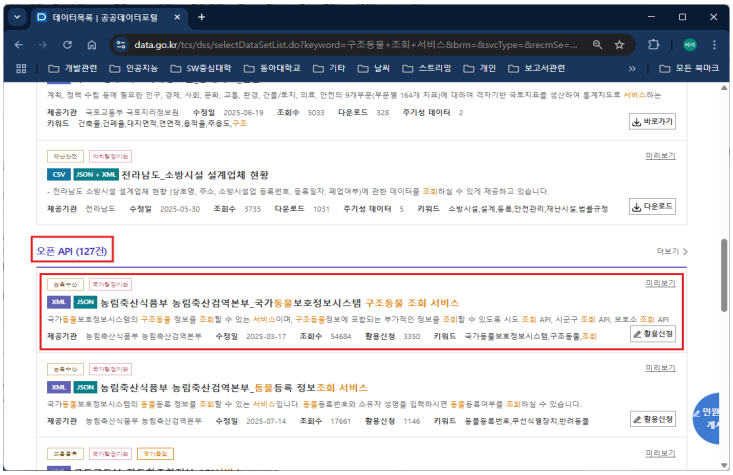
8.2.1 API 검색 및 활용 방법

본 과정은 공공데이터 포털사이트에서 회원가입이 완료된 후 진행할 수가 있습니다. 반드시 앞 절(8.1)에서 진행한 회원가입 및 로그인 과정을 마무리 된 후 진행해 주시기 바랍니다.

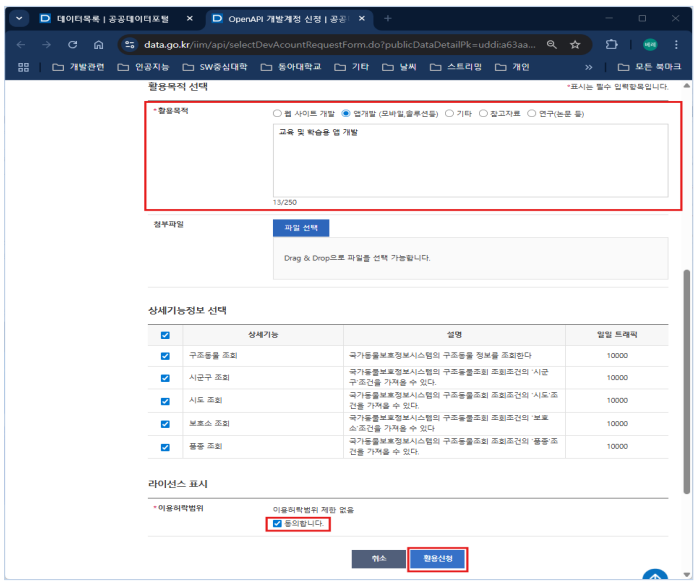
- 공공데이터 포털사이트에서 "구조동물 조회 서비스"
  또는 "국가동물보호정보시스템 구조동물 조회서비스"를 검색합니다.



- 검색 결과 화면을 아래로 스크롤하면 “파일 데이터”와 “오픈 API” 두 가지 유형목록을 확인할 수 있습니다. 본 교재에서는 이 가운데 오픈 API에 해당하는 데이터를 활용하도록 하겠습니다.
  해당 공공데이터의 [활용신청]을 클릭하세요.



- 활용목적→“교육 및 학습용 앱 개발”, 이용허락범위→ “동의합니다.” 체크 등 아래 그림을 참고하여 선택 및 입력한 후 [활용신청] 버튼 클릭하세요.



- 8.1.1 절에서도 언급했듯이 이제 공공데이터를 한 건 신청했으므로, 개인 API인증키를 확인할 수 있을 것입니다. [마이페이지] 클릭하여 개인 API인증키가 발급되었는지 확인해 보세요.
- 인증키는 일반인증키(Encoding)와 디코딩용 인증키(Decoding) 두 가지가 제공 됩니다.
  – 일반인증키 : 웹 브라우저 또는 앱에서 직접 호출할 때 주로 사용
  – 디코딩용 인증키 : 서버에서 데이터를 호출할 때 보안 강화 용도로 사용
- 활용 신청한 공공데이터를 활용한 개발을 위해서는 해당 공공데이터의 필드 구성 및 활용 가이드를 반드시 참고해 두는 것이 좋습니다.

① [API 신청] 클릭하여 신청된 API 리스트 확인

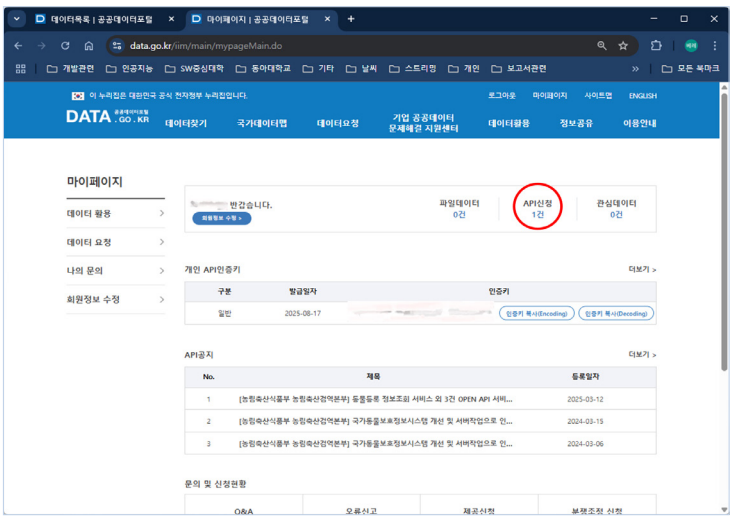
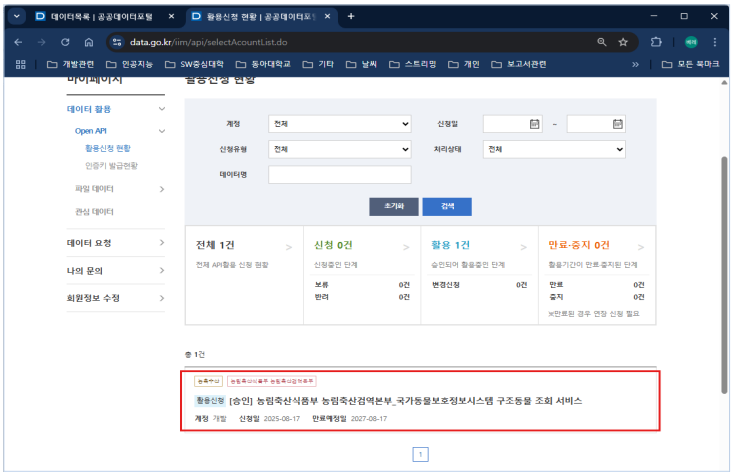


그림 8-1 공공데이터 포털 [마이페이지] 화면



② 활용 가이드 확인

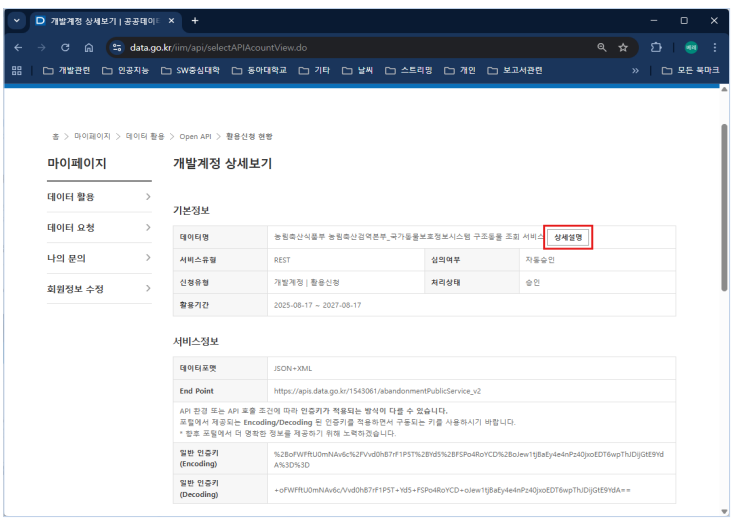
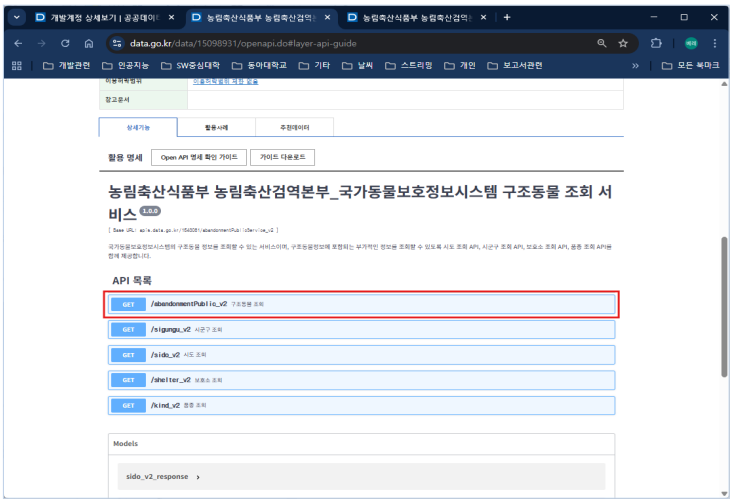


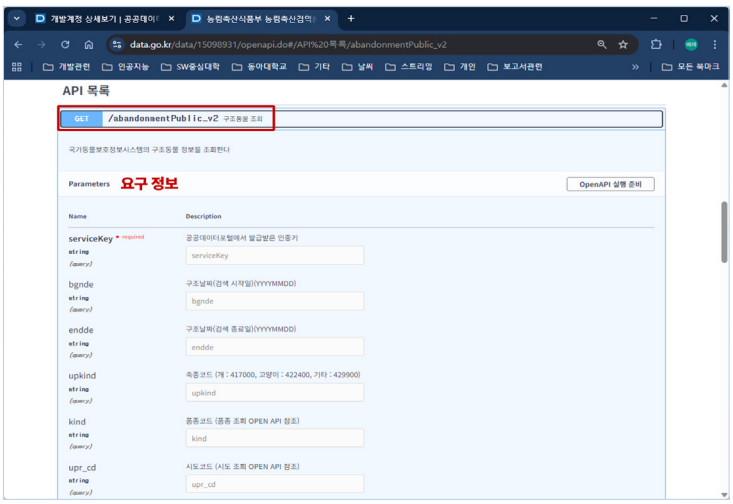
그림 8-2 활용신청 공공데이터 기본정보 화면



해당 API 에서는 5개 상세기능 정보를 포함하고 있습니다. 해당 프로젝트에서는 상세기능 중 “구조동물조회”를 활용할 것이며, 기본 도메인은 “https://apis.data.go .kr/1543061/abandonmentPublicService_v2/” 이고, End Point는 “abandonmentPublic_v2”입니다.

만약 활용할 상세기능 정보가 “시군구조회”일 경우는 기본 도메인은 동일하고, End Point는 “sigungu_v2”가 됩니다.

공공데이터에서 특정 상세 기능의 데이터를 수신하려면, 해당 기능에서 요구하는 정보를 함께 전달해야 합니다. 요구 정보는 활용 가이드에서 확인 가능합니다.



– 공공데이터 API(구조동물조회) 요청 정보(request parameter)

구조동물 조회 API를 호출할 때는, 원하는 조건에 맞는 데이터를 가져오기 위해 다양한 요청 파라미터(request parameter)를 전달해야 한다. 각 파라미터는 검색 조건을 지정하는 역할을 하며, 이를 적절히 활용하면 특정 기간, 지역, 품종, 상태 등의 세부 조건에 따른 데이터를 조회할 수 있다.

표 8-2 구조동물조회(abandonmentPublic_V2) 상세기능 요구정보

Name (필수여부)	설명	입력방식 또는 타입
serviceKey (필수)	발급받은 개인 API 인증키	Enconding key
bgnde	유기동물 구조일 검색 시작일	YYYYMMDD (년도 4자리 + 월 2자리 + 일 2자리)
endde	유기동물 구조일 검색 종료일	YYYYMMDD (년도 4자리 + 월 2자리 + 일 2자리)
upkind	축종 코드	개(417000), 고양이(422400), 기타(429900)
kind	축종별 품종 코드	품종조회 Open API 참조 (/kind_v2)
upr_cd	시군구코드	시군구조회 Open API 참조 (/sigungu_v2)
org_cd	시군구코드	시도조회 Open API 참조 (/sido_v2)
care_reg_no	보호소번호	보호소조회 Open API 참조 (/shelter_v2)
state	상태	notice(공고중), protect(보호중), null
neuter_yn	중성화	Y(예), N(아니오), U(미상)
pageNo (필수)	페이지 번호	숫자
numOfRows (필수)	페이지당 표시할 페이지 개수	숫자
_type (필수)	수신데이터 양식	json, XML
bgupd	수정날짜 검색 시작일	YYYYMMDD (년도 4자리 + 월 2자리 + 일 2자리)
enupd	수정날짜 검색 종료일	YYYYMMDD (년도 4자리 + 월 2자리 + 일 2자리)
sex_cd	성별	M(수컷), F(암컷), Q(미상)
rfid_cd	동물등록번호	숫자
desertion_no	유기번호	
notice_no	공고번호	

– 공공데이터 API(구조동물조회) 응답 정보(response parameter)

공공데이터 포털에서 제공하는 구조동물 조회 API를 호출하면, 요청한 조건에 맞는 구조 · 보호 동물의 세부 정보가 응답으로 전달된다. 응답 데이터는 XML 또는 JSON 형식으로 제공되며, 각 항목은 동물의 기본 정보, 구조 · 공고 정보, 보호소 정보, 이미지 파일 정보 등을 포함한다.

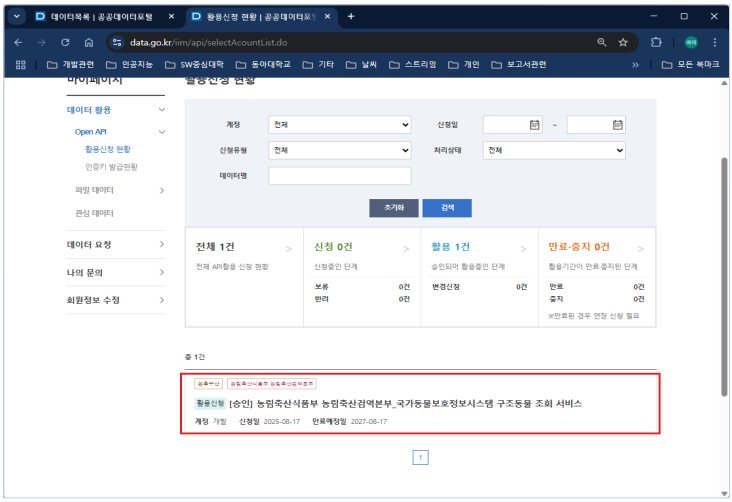
표 8-3 구조동물조회(abandonmentPublic_V2) 응답 정보

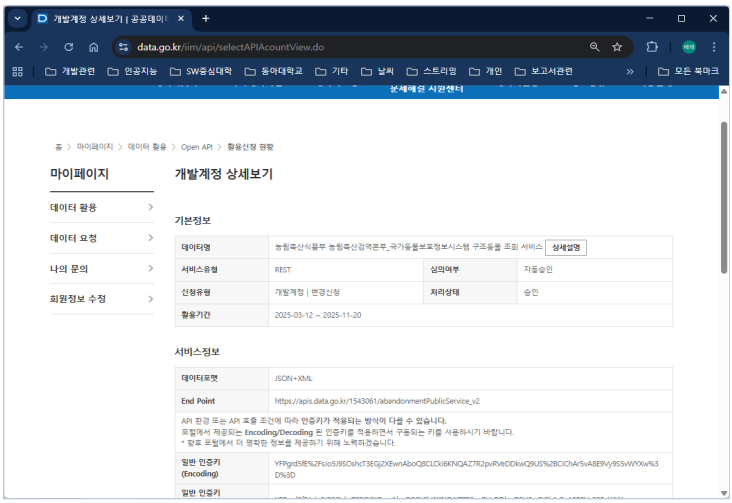
Name	설명	Name	설명
desertionNo	공고번호	popfile2	이미지파일2
happenDt	발견일자	processState	진행상황
happenPlace	발견장소	sexCd	성별
kindFullNm	품종전체이름(예: [개] 푸들)	neuterYn	중성화여부
upKindCd	축종코드	specialMark	특이점
upKindNm	축종명	careRegNo	보호소등록번호
kindNm	품종명(예: 푸들)	careNm	보호소명
colorCd	색상	careTel	보호소 전화번호
age	동물 나이	careAddr	보호소 주소
weight	몸무게	careOwnerNm	보호소 관리자명
noticeNo	공고번호	orgNm	지역
noticeSdt	공고시작일	vaccinationChk	백신접종 내역
noticeEdt	공고종료일	healthChk	감염정보 및 병명
popfile1	이미지파일1	updTm	수정일

구조동물 조회 서비스에서 요청정보로 사용하는 시 · 도 코드(upr_cd), 시 · 군 · 구 코드(org_cd), 그리고 축종별 품종 코드(kind)는 [표 8-2]에서 설명한 것처럼, 해당 코드값을 제공하는 전용 API를 활용하여 조회할 수 있다.

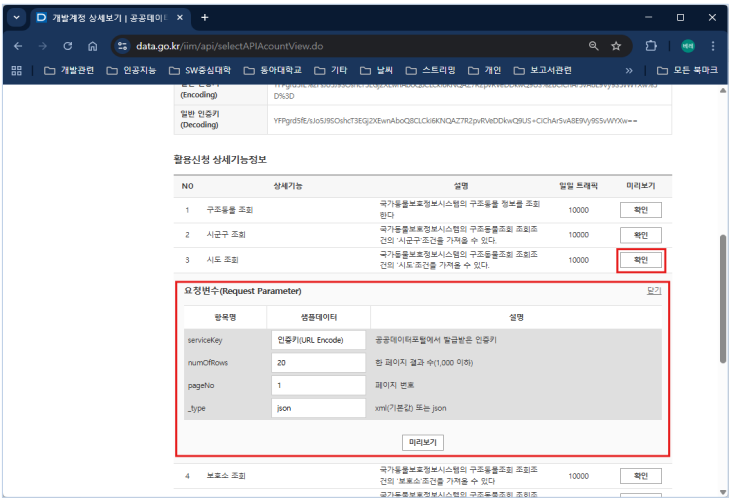
아래는 API를 활용하여 시도 코드(org_cd)를 조회하는 방법입니다.

먼저 공공데이터 마이포털 홈페이지에 접속하여 [마이페이지] 메뉴를 클릭한 뒤, 활용 신청한 API 목록에서 '구조동물조회' API를 선택합니다.

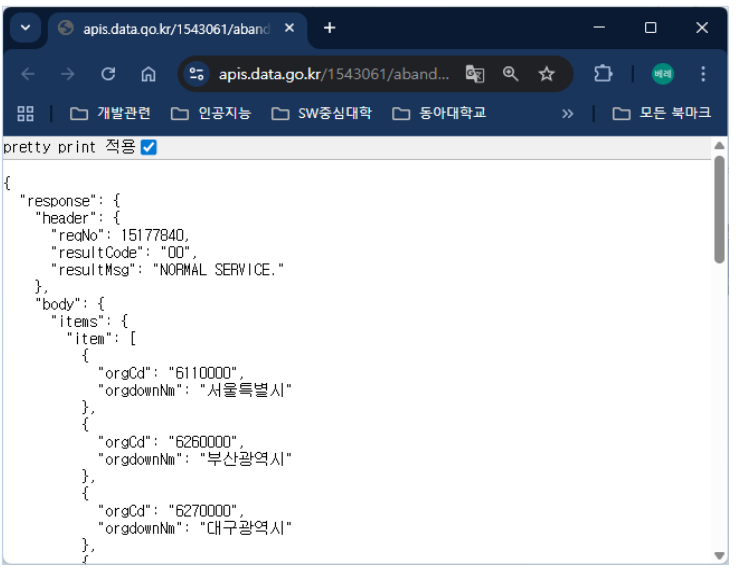




화면을 아래로 스크롤하면 필요한 요청 정보를 입력하고 결과를 직접 테스트할 수 있는 화면이 나타납니다.



이때 '시도 조회' 항목의 [확인] 버튼을 클릭한 후, 미리보기 버튼을 누르면 해당 결과를 확인할 수 있습니다.



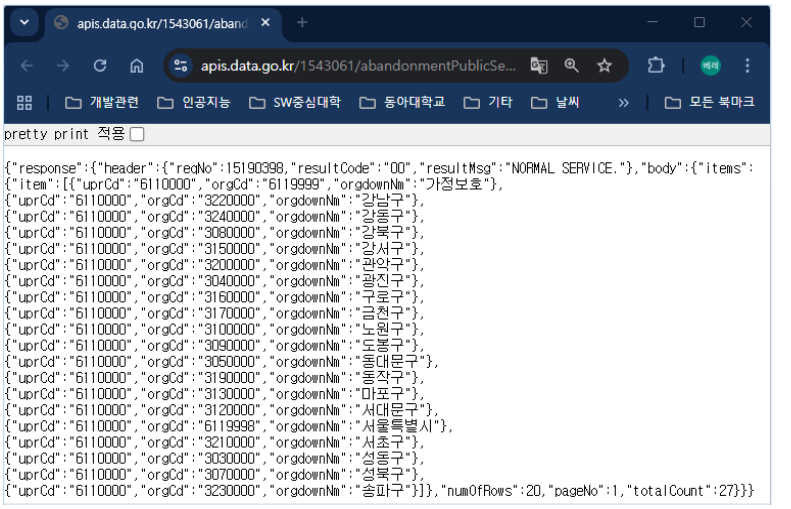
시도조회 결과를 아래에 정리해 두었습니다.

표 8-4 시도조회(sido_v2) 결과

org_cd	설명	org_cd	설명
6110000	서울특별시	6530000	강원특별자치도
6260000	부산광역시	6430000	충청북도
6270000	대구광역시	6440000	충정남도
6280000	인천광역시	6540000	전북특별자치도
6290000	광주광역시	6460000	전라남도
5690000	세종특별자치시	6470000	경상북도
6300000	대전광역시	6480000	경상남도
6310000	울산광역시	650000	제주특별자치도
6410000	경기도		

아래는 시도 코드(org_cd)를 이용해 시군구 조회 입력화면입니다. 화면과 같이 입력하여 결과를 확인해 보세요. serviceKey는 입력하지 않아도 됩니다.





위 방식과 같이 마찬가지로 [표 8-2]에 표기된 축종코드를 이용해 품종코드도 조회해 보길 바랍니다.

8.2.2 공공데이터 활용한 앱 개발

앞 절에서는 공공데이터 포털을 중심으로 데이터를 검색하고 신청 · 활용하는 방법을 익혔지만, 이는 어디까지나 웹 환경에서 제공되는 이용 방식에 국한된 내용이었습니다. 따라서 이번 절에서는 단순히 웹에서 데이터를 확인하는 수준을 넘어, 실제 안드로이드 기반의 어플리케이션을 직접 개발하는 과정을 통해 공공데이터를 어떻게 연동하고 활용할 수 있는지를 프로젝트 실습 형식으로 학습해 보도록 하겠습니다. 이를 통해 학습자는 이론적으로 습득한 공공데이터 활용 지식을 실무적인 모바일 앱 개발 환경에 적용해 보는 경험을 쌓을 수 있습니다.

아래와 같이 설정하고 프로젝트를 생성하세요.

- Project Name : strayAnimal
- Language : Java
- Minimum SDK : API 33 ("Tiramisu"; Android 13.0)

(1) 네트워크 권한 설정 및 프로토콜 승인 (AndroidManifest.xml)

본 프로젝트는 공공데이터를 효과적으로 활용하기 위해, 공공데이터가 저장되어 있는 서버로부터 필요한 정보를 네트워크를 통해 내려받아 처리하는 방식으로 동작합니다. 따라서 애

플리케이션이 정상적으로 실행되기 위해서는 우선 네트워크 접근 권한을 반드시 부여해야 하며, 이를 통해 서버와의 통신이 원활히 이루어질 수 있습니다.

또한 안드로이드 운영체제의 보안 정책상, HTTPS 프로토콜은 별도의 설정 없이 기본적으로 사용할 수 있지만, HTTP 프로토콜은 보안상의 이유로 기본적으로 차단되어 있으며, 이를 사용하기 위해서는 개발자가 명시적으로 설정을 추가해야 합니다. 필자가 본 프로젝트와 관련된 일련의 사항을 검토한 결과, 일부 통신 구간에서 HTTP 프로토콜이 사용되고 있는 것을 확인하였습니다. 따라서 해당 부분이 정상적으로 동작하도록 하기 위해서는 반드시 관련 설정을 추가해 주어야 하며, 이를 통해 애플리케이션이 안정적이고 원활하게 데이터를 주고받을 수 있습니다.

이와 같이 네트워크 접근 권한 부여와 프로토콜 설정은 공공데이터를 기반으로 하는 애플리케이션 개발 과정에서 필수적으로 고려해야 할 요소이며, 이를 간과할 경우 실행 오류나 데이터 통신 실패가 발생할 수 있으므로 주의가 필요합니다.

app 〉 manifests 폴더에 위치한 AndroidManifest.xml 파일을 선택하여 아래와 같이 내용을 수정해 하세요.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <!-- 추가 -->
    <!-- 애플리케이션이 인터넷에 접근할 수 있도록 권한 부여 -->
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.StrayAnimal"
      <!-- 추가 -->
      <!-- http(암호화되지 않은 트래픽) 허용 여부 -->
        android:usesCleartextTraffic="true"
```

```
        <!-- 네트워크 보안설정 파일 지정  -->
        android:networkSecurityConfig="@xml/network_security_config">

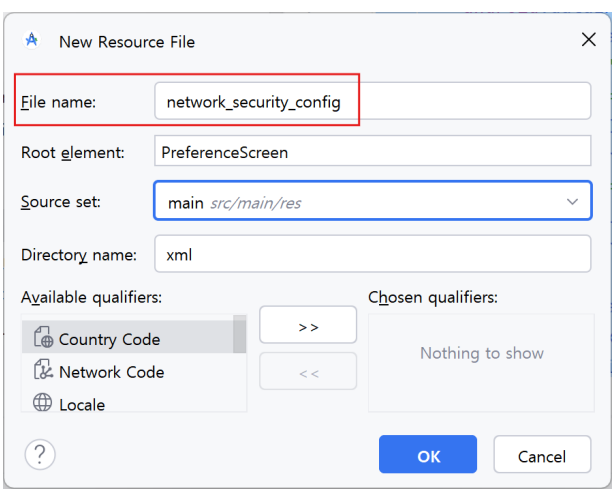
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

위 수정된 파일에서 @xml/network_security_config 부분을 확인할 수 있을 것입니다.

해당 파일도 직접 작성해야 하므로 아래 단계를 따라 작성해 주세요.

- network_security_config 파일 생성:
  - app 〉 res 〉 xml 폴더에서 마우스 우클릭
  - New 〉 XML Resource File 선택
  - File name → network_security_config 입력



파일 생성 후, [Code] 모드로 변경하여 다음과 같이 코드를 입력하세요.

network_security_config.xml

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <!-- 필요한 도메인만 추가 -->
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">openapi.animal.go.kr</domain>
    </domain-config>
</network-security-config>
```

위 코드에서 사용된 설정은 "openapi.animal.go.kr" 이라는 도메인이 http 프로토콜을 사용하는 사이트이므로, 해당 사이트에 대해 암호화되지 않은 통신을 허용하겠다는 의미를 담고 있습니다. 기본적으로 안드로이드에서는 보안 강화를 위해 https가 아닌 http 요청을 차단하지만, network_security_config 파일을 통해 특정 도메인에 한해서 예외적으로 허용할 수 있습니다.

따라서 현재 설정은 "openapi.animal.go.kr"을 대상으로만 http 접근을 허용하도록 지정되어 있습니다. 만약 이와 같은 방식으로 다른 URL을 추가적으로 허용해야 할 경우에는, 〈domain includeSubdomains="true"〉〈/domain〉 태그 사이에 허용하려는 도메인 주소를 추가하면 됩니다. 이렇게 하면 지정한 도메인뿐만 아니라 해당 도메인의 하위 서브도메인들까지 모두 http 프로토콜로 접근할 수 있게 되므로, 필요한 경우 유연하게 확장할 수 있습니다.

즉, 개발자가 활용하고자 하는 공공데이터 API나 특정 서버가 http 기반으로만 제공되는 상황에서는, 보안 설정을 통해 명시적으로 허용 도메인을 지정해야 애플리케이션이 정상적으로 데이터를 주고받을 수 있다는 점을 이해하는 것이 중요합니다.

(2) 화면 구성 (activity_main.xml)

메인 화면 구성을 위해 메인화면 디자인 모드를 [Code] 모드로 변경하고 아래 코드를 입력하세요.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:padding="16dp"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <LinearLayout
        android:orientation="horizontal"
```

```
        android:gravity="center_vertical"
        android:layout_width="match_parent" android:layout_height="wrap_content">

        <Button
            android:id="@+id/btn_pick_dates"
            android:text="검색 기간 선택"
            android:layout_width="wrap_content" android:layout_height="wrap_content"/>

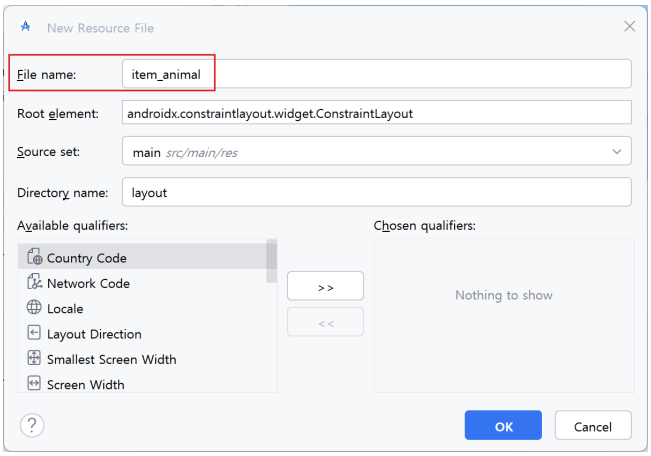
        <TextView
            android:id="@+id/tv_period"
            android:text="기간: -"
            android:layout_marginStart="12dp"
            android:layout_width="wrap_content" android:layout_height="wrap_content"/>
    </LinearLayout>

    <ListView
        android:id="@+id/lv_animals"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="0.5dp"
        android:layout_width="match_parent" android:layout_height="0dp"
        android:layout_weight="1"/>
</LinearLayout>
```

(3) 아이템 레이아웃 (item_animal.xml)

리스트의 각 항목을 정의하는 새로운 레이아웃 파일을 생성해야 합니다.

- 아이템 레이아웃 파일 생성 :
  - app 〉 res 〉 layout 폴더에서 마우스 우클릭
  - New 〉 Layout Resource File 선택
  - File name → item_animal 입력



생성된 파일을 열어, [Code] 모드로 변경하여 아래 코드를 입력하세요.

item_animal.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:padding="12dp"
    android:layout_width="match_parent" android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/img_photo"
        android:layout_width="80dp" android:layout_height="80dp"
        android:scaleType="centerCrop"
        android:src="@mipmap/ic_launcher"/>

        <LinearLayout
            android:orientation="vertical"
            android:layout_marginStart="12dp"
                android:layout_width="0dp" android:layout_height="wrap_content"
                android:layout_weight="1">

            <TextView
                android:id="@+id/tv_kind"
                android:textStyle="bold"
                android:textSize="16sp"
                android:layout_width="wrap_content" android:layout_height="wrap_
content"/>

            <TextView
```

```
            android:id="@+id/tv_careNm"
            android:textSize="14sp"
            android:layout_width="wrap_content" android:layout_height="wrap_content"/>
    </LinearLayout>
</LinearLayout>
```

(4) 데이터 모델 정의

구조동물 조회 서비스에서 제공하는 응답 데이터의 유형은 JSON과 XML 두 가지 형식으로 나뉘어 있으며, 사용자는 필요에 따라 원하는 형식을 선택하여 활용할 수 있습니다. 본 프로젝트에서는 그 중 JSON 형식을 선택하여 서버로부터 데이터를 수신하고 처리하도록 하였습니다.

특히, [표 8-3]에 제시된 응답 정보에는 매우 다양한 항목이 포함되어 있지만, 실제 애플리케이션 구현 과정에서 모든 데이터를 활용할 필요는 없습니다. 따라서 전체 데이터를 그대로 처리하기보다는, 앱의 기능 구현에 직접적으로 필요한 항목만을 신중하게 선별하여 매핑할 계획입니다.

이와 같은 방식은 불필요한 데이터를 모두 다루는 과정에서 발생할 수 있는 메모리 낭비와 처리 속도 저하를 방지하고, 동시에 유지보수의 편의성을 높여줍니다. 결과적으로, 프로젝트에서는 필요한 데이터만 선별적으로 활용함으로써 보다 효율적이고 최적화된 데이터 관리 및 처리가 가능해질 것입니다.

해당 부분에서 작성하는 데이터 모델 파일은 서버로부터 전달받은 응답 데이터를 저장하고 관리하기 위한 일종의 틀, 즉 그릇 역할을 하는 클래스라고 이해할 수 있습니다. 이 클래스는 공공데이터 API에서 내려오는 여러 항목 중 필요한 데이터들을 속성(필드)으로 정의하고, 이를 객체 형태로 다룰 수 있도록 해줍니다.

즉, 단순히 데이터를 수신하는 데 그치지 않고, 애플리케이션 내부에서 해당 데이터를 보다 구조적이고 체계적으로 활용할 수 있도록 도와주는 역할을 수행합니다. 이렇게 정의된 데이터 모델 클래스는 이후 어댑터나 뷰와 연결되어 화면에 표시되거나, 추가적인 로직에서 가공 및 처리될 수 있으므로, 앱 개발 과정에서 핵심적인 기반 요소라고 할 수 있습니다.

또한, 한 가지 짚고 넘어가야 할 점은 프로젝트 내 파일 구성 방식입니다. 기존 본 교재에서 다루었던 예제 프로젝트에서는 모든 Java 파일을 단일 패키지, 즉 com.example.projectname 이

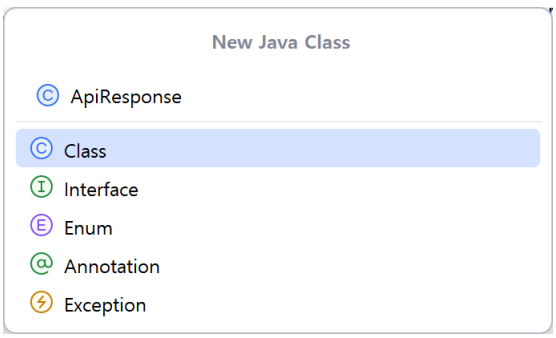
라는 하나의 폴더 안에 모두 생성하는 방식으로 진행되었습니다. 이러한 방식은 초보자가 프로젝트 구조를 단순하게 이해하는 데에는 도움이 될 수 있으나, 실제 개발 현장에서는 권장되지 않는 방식입니다. 왜냐하면 프로젝트가 점점 커지고 기능이 다양해질수록, 모든 파일이 한 폴더에 뒤섞여 있으면 파일 관리가 어려워지고, 가독성이 떨어지며, 유지보수나 확장이 비효율적이 되기 때문입니다. 따라서 올바른 접근 방식은, 기능별로 별도의 패키지(폴더)를 생성하고, 관련 있는 파일들을 묶어서 배치하는 것입니다. 예를 들어, 네트워크 통신과 관련된 파일은 data 패키지에, 화면 UI와 관련된 파일은 ui 패키지에, 그리고 데이터 모델 클래스는 model 패키지에 두는 식으로 구조를 나누는 것이 바람직합니다. 이렇게 하면 프로젝트의 전반적인 구조가 한눈에 들어올 뿐만 아니라, 협업 시에도 각 기능의 위치를 쉽게 파악할 수 있으며, 코드 수정이나 기능 확장 시에도 혼란을 줄일 수 있습니다.

결국, 패키지를 기능 단위로 세분화하여 구성하는 것은 프로젝트의 체계성과 유지보수성, 그리고 확장성을 동시에 확보하는 가장 올바른 방법이라고 할 수 있습니다.

- Model 폴더 생성 :
  - app 〉 java 〉 com.example.strayanimal 폴더에서 마우스 우클릭
  - New 〉 Package 클릭
  - com.example.strayanimal.model 입력

(com.example.strayanimal. 은 기본으로 입력되어 있으므로 model만 작성하면 됩니다.)

- Model 파일 생성
  - app 〉 java 〉 com.example.strayanimal 〉 model 폴더에서
  - new 〉 Java Class 클릭
  - class 명은 “ApiResponse”로 입력



아래와 같이 코드를 입력하세요.

ApiResponse.java

```
package com.example.strayanimal.model;

// 공공데이터API 응답을 담는 데이터 모델 클래스
public class ApiResponse {
    // 최상위 응답 객체
    public Response response;

    // 응답(response) 내부 구조
    public static class Response {
        public Body body; // 실제 데이터(body) 부분
    }

    // body 영역 구조
    public static class Body {
        public Items items;     // 조회된 동물 정보 목록
        public int totalCount;  // 전체 데이터 개수
        public int pageNo;      // 현재 페이지 번호
        public int numOfRows;   // 한 페이지당 출력되는 행(row)의 수
    }

    // items 영역 구조
    public static class Items {
        public java.util.List<Item> item; // 실제 동물 정보 리스트
    }

    // 실제 구조동물 한 마리의 상세 정보
    public static class Item {
        public String popfile1;    // 동물 이미지URL
        public String kindFullNm;  // 품종 정보(예: "[개] 믹스견")
        public String careNm;      // 보호소명
    }
}
```

위의 코드는 표 8-3에 제시된 수신 데이터 구조를 참조하여 작성된 것으로, 응답 데이터 전체를 모두 포함한 것이 아니라 실제 프로젝트 구현에 필요한 항목만 선별하여 구성하였습니다. 다시 말해, 불필요한 데이터를 모두 담는 대신 앱 개발 과정에서 반드시 필요한 정보만을 추출하여 데이터 모델에 정의한 것입니다.

특히 public static class Item { … } 부분에 선언된 popfile1, kindFullNm, careNm 변수명은 [표

8-3]에서 제시된 원본 응답 데이터의 항목명과 동일하게 맞추어져 있습니다. 이는 JSON 데이터를 매핑할 때 Gson과 같은 라이브러리가 자동으로 해당 이름을 인식하여 정확하게 변환할 수 있도록 하기 위함입니다. 따라서 여러분이 이 외의 다른 항목을 추가하여 활용하고자 한다면, 반드시 표 8-3에 나와 있는 항목명을 그대로 사용하여 변수명을 정의해야 합니다.

아울러 이 변수명들은 이후 데이터가 실제 화면에 표시되는 과정에서 Adapter 클래스를 통해 활용되므로, 모델 클래스에서 변수명을 변경하면 Adapter에서도 동일하게 수정해 주어야 합니다. 만약 두 부분의 변수명이 일치하지 않으면 데이터가 정상적으로 매핑되지 않아 화면에 올바르게 표시되지 않는 문제가 발생할 수 있으므로, 반드시 주의해야 합니다. Adapter 파일 작성시 한번 더 언급하도록 하겠습니다.

즉, 데이터 모델과 Adapter는 긴밀하게 연결되어 있기 때문에, 새로운 항목을 추가하거나 기존 변수명을 변경할 경우에는 항상 표 8-3의 항목명과의 일치 여부를 확인하고, 동시에 Adapter에서도 동일하게 반영해 주는 것이 중요합니다.

아래 그림은 구조동물 조회 서비스 API 응답 데이터의 계층 구조를 시각적으로 도식화한 다이어그램으로, 실제 API에서 전달되는 JSON 데이터가 어떤 형태로 중첩되고 배열되는지를 한눈에 확인할 수 있도록 표현한 것입니다. 다이어그램을 살펴보면 response를 최상위 루트로 하여, 그 안에 header, body, 그리고 items, 마지막으로 여러 개의 item들이 배열 형태로 포함되어 있음을 알 수 있습니다. 각 item에는 유기번호(desertionNo), 보호소명(careNm), 품종명(kindNm), 이미지 파일 경로(popfile1) 등과 같은 세부 항목이 담겨 있습니다.

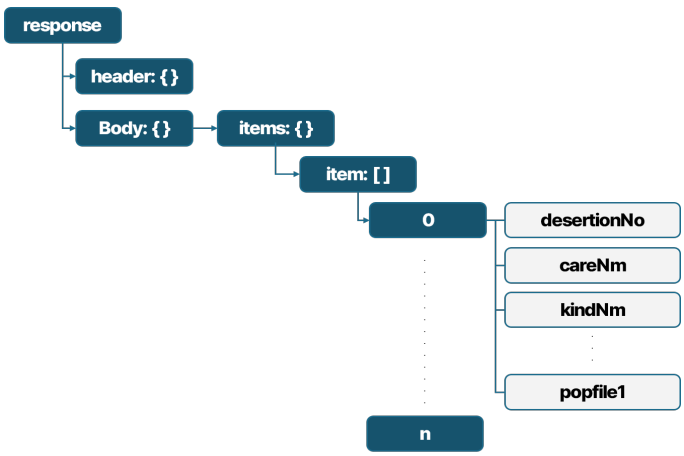


그림 8-2 구조동물조회 서비스 API 응답 데이터 계층 구조

이 다이어그램을 앞서 작성한 ApiResponse.java 파일과 비교해 보시면, 왜 Api Response 클래스가 response → body → items → item 구조로 중첩 클래스를 통해 정의되어 있는지를 쉽게 이해할 수 있습니다. 즉, 다이어그램의 계층 구조를 그대로 코드로 옮겨 놓은 것이 바로 ApiResponse.java이며, 이를 통해 JSON 응답을 자바 객체로 자동 매핑하여 보다 효율적으로 다룰 수 있게 되는 것입니다.

따라서 이 다이어그램은 ApiResponse.java의 개념을 이해하는 데 중요한 시각적 단서가 되며, 실제 응답 데이터 구조와 코드 간의 일대일 대응 관계를 명확하게 보여주는 자료라고 할 수 있습니다.

(5) 요청 정보 정의 (Retrofit 인터페이스)

안드로이드에서 Retrofit을 사용하려면, 단순히 라이브러리를 추가하는 것만으로는 서버와 통신할 수 없습니다. 서버의 엔드포인트(End Point), 즉 어떤 주소(URL)로 어떤 방식(GET, POST 등)으로 요청을 보낼지, 그리고 어떤 데이터 형식(JSON, XML 등)으로 받을지를 정의해야 합니다.

Retrofit은 이런 요청 정보를 인터페이스(Interface) 형태로 선언하게끔 설계되어 있습니다.

즉, 개발자가 직접 네트워크 요청 코드를 작성하는 것이 아니라, 인터페이스에 메서드를 정의하면 Retrofit이 내부적으로 실제 HTTP 요청 코드를 자동으로 생성해주는 방식입니다. 이렇게 함으로써 코드가 훨씬 간결해지고, 가독성이 높아지며, API 변경 시에도 수정해야 할 부분이 명확해집니다.

- Retrofit 폴더 생성 :
  - app 〉 java 〉 com.example.strayanimal 폴더에서 마우스 우클릭
  - New 〉 Package 클릭
  - com.example.strayanimal.data 입력
- Retrofit 파일 생성 :
  - app 〉 java 〉 com.example.strayanimal 〉 data 폴더에서 마우스 우클릭
  - New 〉 Java Class를 선택하고,
  - class 명은 “ApiClient.java”로 입력

ApiClient.java

```
package com.example.strayanimal.data;

import com.example.strayanimal.model.ApiResponse;
import retrofit2.Call;
import retrofit2.http.GET;
import retrofit2.http.Query;

// Retrofit을 통해API 요청을 정의하는 인터페이스
public interface ApiClient {

// HTTP GET 방식으로 요청
// 요청URL의 기본 구조는BASE_URL + "abandonmentPublicService_v2/abandonmentPublic_v2"
   @GET("abandonmentPublicService_v2/abandonmentPublic_v2")
   Call<ApiResponse> getAnimals(

   // API 인증키
   // 공공데이터포털에서 발급받은 서비스 키(URL 인코딩된 값 그대로 사용)
      @Query(value="serviceKey", encoded = true) String serviceKey,

   // 응답 형식 지정
   // json 형식으로 받기 위해"_type"에"json" 입력
      @Query("_type") String type,

   // 한 페이지에 가져올 데이터 개수
   // ex) 10 → 한 번 요청 시10개의 동물 정보 수신
      @Query("numOfRows") int numOfRows,

   // 현재 페이지 번호
   // ex) 1 → 첫 번째 페이지 데이터 요청
      @Query("pageNo") int pageNo,

   // 축종 코드
   // 417000 = 개(유기견), 422400 = 고양이, 429900 = 기타
      @Query("upkind") String upkind,

   // 검색 시작일
   // 조회 기간의 시작 날짜(형식: yyyyMMdd)
      @Query("bgnde") String bgnde,

   // 검색 종료일
   // 조회 기간의 종료 날짜(형식: yyyyMMdd)
      @Query("endde") String endde,

   // 동물 상태(선택적 파라미터)
```

```
        // "notice" = 공고중, "protect" = 보호중
        @Query("state") String state
    );
}
```



코드 설명

① @GET("abandonmentPublicService_v2/abandonmentPublic_v2")
→ API 요청을 GET 으로 지정
→ @GET("abandonmentPublicService_v2/abandonmentPublic_v2")는 End Point 경로이며,
Retrofit.Builder()에서 지정한 BASE_URL과 합쳐져 최종 요청주소가 완성됩니다.

② Call<ApiResponse> getAnimals(.... 이하 생략
→ API 호출 메서드를 정의한 부분
→ 반환 타입은 ApiResponse 로 지정되어 있으며, 이는 API 호출 결과가 ApiResponse 모델 클래스 구조에 맞게 자동 매핑됩니다.

③ @Query("_type") String type,
→ @Query 어노테이션이라고 칭함.
→ 정의된 변수 type 에 데이터가 담겨져 나중에 Retrofit에 지정된 최종 요청주소와 함께 서버에 전송된다

(6) 네트워크 모듈

네트워크 모듈(Network Module) 은 ApiClient 같은 Retrofit 인터페이스를 실제로 동작할 수 있도록 초기화하고 관리하는 역할을 하는 클래스입니다.

즉, Retrofit을 직접 생성하고, 기본 URL, Gson 변환기, OkHttpClient(로그 출력, 타임아웃, 헤더 설정 등)를 설정해 API 호출 객체를 만들어 주는 부분을 네트워크 모듈이라고 합니다.

- Network 모듈 파일 생성 :
  - app 〉 java 〉 com.example.strayanimal 〉 data 폴더에서 마우스 우클릭
  - New 〉 Java Class 클릭
  - class 명은 "NetworkModule"로 입력

NetworkModule.java

```
package com.example.strayanimal.data;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

import okhttp3.OkHttpClient;
import okhttp3.logging.HttpLoggingInterceptor;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class NetworkModule {
    // 공공데이터포털 동물보호 관련API의 공통BASE URL
    // 실제 요청URL은ApiClient의@GET 경로가 뒤에 붙어 완성됩니다.
    // 예: https://apis.data.go.kr/1543061/ +
                        abandonmentPublicService_v2/abandonmentPublic_v2
    private static final String BASE_URL = "https://apis.data.go.kr/1543061/";

    // Retrofit 인스턴스를 생성해 반환
    public static Retrofit provideRetrofit() {
        // 네트워크 요청/응답 로그 확인을 위한 인터셉터(개발/디버그 시 유용)
        HttpLoggingInterceptor logger = new HttpLoggingInterceptor();
            logger.setLevel(HttpLoggingInterceptor.Level.BODY); // 헤더/바디까지 전체 로그

            // OkHttp 클라이언트 구성: 인터셉터 등 네트워크 관련 옵션을 부착
            OkHttpClient ok = new OkHttpClient.Builder()
                            .addInterceptor(logger) // 네트워크 로그 출력
                            .build();

            // Retrofit 빌더: BASE_URL, OkHttpClient, Converter(Gson) 설정
            return new Retrofit.Builder()
                    .baseUrl(BASE_URL)                  // 기본URL 설정
                    .client(ok)                         // 위에서 구성한OkHttpClient 적용
                      // JSON ↔ 객체 변환을 위한Gson 컨버터
                    .addConverterFactory(GsonConverterFactory.create())
                      .build();
    }

// ApiClient 구현체 생성 헬퍼: 어디서든NetworkModule.api()로 사용 가능
    public static ApiClient api() {
        return provideRetrofit().create(ApiClient.class);
    }
}
```

(7) 어댑터 정의

7장에서 살펴보았던 어댑터에서 언급한 것처럼 어댑터는 데이터 항목을 어떻게 표현할지를 정의하는 부분입니다. 어댑터는 데이터를 배치한다는 의미로 ui 개념과 유사하다. 따라서 아래와 같이 ui 폴더를 생성하여 그 아래에 파일을 생성합니다.

- ui 폴더 생성 :
  - app 〉 java 〉 com.example.strayanimal 폴더에서 마우스 우클릭
  - New 〉 Package 클릭
  - com.example.strayanimal.ui 입력
- Adapter 파일 생성 :
  - app 〉 java 〉 com.example.strayanimal 〉ui 폴더에서 마우스 우클릭
  - New 〉 Java Class를 클릭
  - class 명은 "animalAdapter"로 입력

AnimalAdapter.java

```
package com.example.strayanimal.ui;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import com.bumptech.glide.Glide;
import com.example.strayanimal.R;
import com.example.strayanimal.model.ApiResponse;

import java.util.ArrayList;
import java.util.List;

// 구조동물 데이터를ListView에 표시하기 위한Custom Adapter
public class AnimalAdapter extends BaseAdapter {
    private final Context context;                    // UI 컨텍스트(Activity)
    private final List<ApiResponse.Item> data = new ArrayList<>(); // 표시할 데이터 목록

    // 생성자: Adapter 생성 시Context 전달받음
```

```
    public AnimalAdapter(Context context) {
        this.context = context;
    }

    // 외부에서 데이터를 갱신할 때 사용하는 메서드
    public void setItems(List<ApiResponse.Item> items) {
        data.clear();              // 기존 데이터 삭제
        if (items != null) data.addAll(items); // 새로운 데이터 추가
            notifyDataSetChanged();   // ListView에 데이터 변경 알림→ 화면 갱신
    }

    // 전체 아이템 개수 반환
    @Override
    public int getCount() {
        return data.size();
    }

    // 특정 위치(position)의 아이템 반환
    @Override
    public ApiResponse.Item getItem(int position) {
        return data.get(position);
    }

    // 아이템 고유ID 반환(여기서는position 값 그대로 사용)
    @Override
    public long getItemId(int position) {
        return position;
    }

    // ViewHolder 패턴: findViewById 반복 호출을 줄여 성능 최적화
    static class ViewHolder {
        ImageView imgPhoto; // 동물 사진
        TextView tvKind;    // 품종 정보
        TextView tvCareNm;  // 보호소 이름
    }

    // 실제로 각 아이템 뷰를 생성하거나 재활용하는 부분
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder h;

        // convertView가null이면 새로운 뷰를 생성하고ViewHolder에 바인딩
        if (convertView == null) {
            convertView = LayoutInflater.from(context).inflate(R.layout.item_animal, parent, false);
            h = new ViewHolder();
            h.imgPhoto = convertView.findViewById(R.id.img_photo);
            h.tvKind   = convertView.findViewById(R.id.tv_kind);
```

```
            h.tvCareNm = convertView.findViewById(R.id.tv_careNm);
            convertView.setTag(h); // ViewHolder 객체를View에 저장
        } else {
            // 기존 뷰가 있으면 재활용→ ViewHolder 불러오기
            h = (ViewHolder) convertView.getTag();
        }

        // 현재 위치의 데이터 가져오기
        ApiResponse.Item item = getItem(position);

        // 품종과 보호소명 설정(null일 경우"-" 표시)
        h.tvKind.setText(item.kindFullNm != null ? item.kindFullNm : "-");
        h.tvCareNm.setText(item.careNm != null ? item.careNm : "-");

    // Glide 라이브러리를 이용해 이미지URL 로드(비동기 로딩)
        Glide.with(context)
            .load(item.popfile1)                    // 이미지URL
            .placeholder(R.mipmap.ic_launcher)      // 로딩 중일 때 표시할 기본 이미지
            .centerCrop()                           // 이미지 크기 조정
            .into(h.imgPhoto);                      // ImageView에 삽입

        return convertView; // 최종View 반환
    }
}
```

(8) 메인코드 작성

공공데이터 어플리케이션 개발을 위해 지금까지 모델(Model) 구성, 레이아웃(Layout) 설계, 그리고 어댑터(Adapter) 구현 과정을 단계별로 완료하였습니다. 이러한 준비 과정을 통해 데이터의 구조가 체계적으로 정의되고, 화면에 표시될 사용자 인터페이스가 설계되었으며, 데이터와 화면 요소를 연결해 주는 어댑터 또한 준비된 상태입니다. 이제 이와 같이 구성된 요소들을 바탕으로 실제 동작하는 애플리케이션을 구현하기 위해 메인 코드 작성 단계로 나아가도록 하겠습니다.

MainActivity.java

```
package com.example.strayanimal;

import androidx.appcompat.app.AppCompatActivity;
import android.app.DatePickerDialog;
import android.os.Bundle;
```

```
import android.util.Log;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import com.example.strayanimal.data.NetworkModule;
import com.example.strayanimal.data.ApiClient;
import com.example.strayanimal.model.ApiResponse;
import com.example.strayanimal.ui.AnimalAdapter;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Locale;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

// 메인 화면Activity: 유기동물 데이터를 조회하고ListView로 출력
public class MainActivity extends AppCompatActivity {

    // UI 컴포넌트
    private Button btnPickDates;   // 날짜 선택 버튼
    private TextView tvPeriod;     // 선택된 기간 표시
    private ListView lvAnimals;    // 동물 리스트 표시할ListView
    private AnimalAdapter adapter; // Custom Adapter

    // API 요청용 날짜 변수(yyyyMMdd 형식)
    private String bgnde; // 검색 시작일
    private String endde; // 검색 종료일

    // 날짜 포맷터
    private final SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd",
                                                              Locale.KOREA); // API용
    private final SimpleDateFormat sdfUi = new SimpleDateFormat("yyyy.MM.dd",
                                                              Locale.KOREA); // 화면 표시용

    // TODO: 본인 서비스 키로 교체 필요
    private static final String SERVICE_KEY = "발급받은 개인 인증키";
    private static final String UPKIND_DOG = "417000"; // 개(유기견) 코드

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

```
        // UI 요소 초기화
        btnPickDates = findViewById(R.id.btn_pick_dates);
        tvPeriod     = findViewById(R.id.tv_period);
        lvAnimals    = findViewById(R.id.lv_animals);

        // Adapter 연결
        adapter = new AnimalAdapter(this);
        lvAnimals.setAdapter(adapter);

        // 초기값: 오늘 날짜~ 오늘 날짜
        Calendar cal = Calendar.getInstance();
        endde = sdf.format(cal.getTime());
        bgnde = sdf.format(cal.getTime());
        tvPeriod.setText("기간: " + sdfUi.format(cal.getTime()) + " ~ " +
                                                        sdfUi.format(cal.getTime()));

        // 버튼 클릭 시 시작일 선택→ 이후 종료일 선택
        btnPickDates.setOnClickListener(v -> pickStartDate());

        // 앱 최초 실행 시 데이터 요청
        requestAnimals();
    }

    // 시작 날짜 선택 다이얼로그
    private void pickStartDate() {
        final Calendar c = Calendar.getInstance();
        DatePickerDialog dpd = new DatePickerDialog(
                                    this,  (view, year, month, dayOfMonth) -> {
            Calendar s = Calendar.getInstance();
            s.set(year, month, dayOfMonth, 0, 0, 0);
            bgnde = sdf.format(s.getTime()); // 선택된 날짜를 검색 시작일로 저장
            pickEndDate(); // 이어서 종료일 선택
        }, c.get(Calendar.YEAR), c.get(Calendar.MONTH), c.get(Calendar.DAY_OF_MONTH)
    );
    dpd.setTitle("검색 시작일 선택");
    dpd.show();
}

    // 종료 날짜 선택 다이얼로그
    private void pickEndDate() {
        final Calendar c = Calendar.getInstance();
        DatePickerDialog dpd = new DatePickerDialog(
                        this, (view, year, month, dayOfMonth) -> {
            Calendar e = Calendar.getInstance();
            e.set(year, month, dayOfMonth, 23, 59, 59);
            endde = sdf.format(e.getTime()); // 선택된 날짜를 검색 종료일로 저장
```

```
            // UI에 선택된 기간 표시
            tvPeriod.setText("기간: " + toUi(bgnde) + " ~ " + toUi(endde));
            // 새로운 기간으로API 재요청
            requestAnimals();
        }, c.get(Calendar.YEAR), c.get(Calendar.MONTH), c.get(Calendar.DAY_OF_MONTH)
    );
    dpd.setTitle("검색 종료일 선택");
    dpd.show();
    }

    // yyyyMMdd → yyyy.MM.dd 변환(UI 출력용)
    private String toUi(String ymd) {
        try {
            return sdfUi.format(
                    new SimpleDateFormat("yyyyMMdd", Locale.KOREA).parse(ymd));
        } catch (Exception e) { return ymd; }
    }

    // 유기동물API 요청 메서드
    private void requestAnimals() {
        // Retrofit ApiClient 생성
        ApiClient api = NetworkModule.api();

        // API 요청 정의
        Call<ApiResponse> call = api.getAnimals(
            SERVICE_KEY,
            "json",      // 응답 형식
            30,          // 한 페이지 조회 개수
            1,           // 페이지 번호
            UPKIND_DOG,  // 축종: 유기견
            bgnde,       // 검색 시작일
            endde,       // 검색 종료일
            ""           // 상태: 전체(원하면"notice" 또는 "protect" 가능)
        );

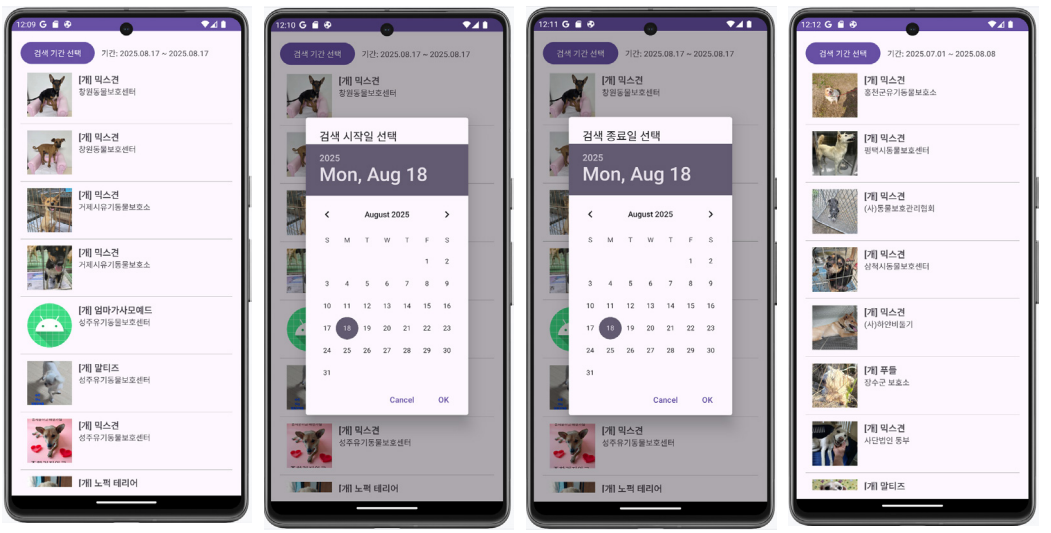
        // 비동기 요청 실행
        call.enqueue(new Callback<ApiResponse>() {
            @Override
            public void onResponse(Call<ApiResponse> call, Response<ApiResponse> res) {
                // 응답이 성공적이고 필요한 데이터가null이 아닐 경우
                if (res.isSuccessful() && res.body() != null &&
                    res.body().response != null &&
                    res.body().response.body != null &&
                    res.body().response.body.items != null) {

                        // Adapter에 데이터 설정 후UI 갱신
```

```
                    adapter.setItems(res.body().response.body.items.item);

                    // 결과 개수 토스트 출력
                    int count = (res.body().response.body.items.item == null) ? 0
                                        : res.body().response.body.items.item.size();
                    Toast.makeText(MainActivity.this, "결과: " + count + "건",
                                                Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(MainActivity.this, "응답 파싱 실패",
    Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<ApiResponse> call, Throwable t) {
                // 네트워크 요청 실패 시 메시지 표시
                Toast.makeText(MainActivity.this, "요청 실패: " + t.getMessage(),
                                                Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```



위 프로그램은 실행과 동시에 현재 날짜를 검색 시작일과 종료일로 자동 설정하여, 해당 기간에 해당하는 유기동물 정보를 불러와 리스트 형태로 화면에 표시해 줍니다. 또한 사용자가

[검색기간 선택] 버튼을 클릭하면, 시작일과 종료일을 직접 지정할 수 있도록 날짜 선택 대화상자(DatePickerDialog)가 제공되며, 사용자가 선택한 기간에 맞추어 다시 데이터를 조회하여 새로운 결과를 화면에 갱신해 주는 구조로 동작합니다.

이 프로그램은 기본적으로 날짜 범위를 기반으로 한 검색 기능을 중심으로 구현되어 있지만, 지금까지 학습한 내용을 응용하여 표시되는 항목을 변경해 보거나, 출력되는 데이터 형식을 다양하게 가공해 보는 것도 좋은 학습 방법이 될 수 있습니다. 예를 들어, 현재는 보호소 이름과 품종, 이미지 정도만 보여주고 있지만, 구조 일자나 발견 장소와 같은 추가 항목을 함께 표시하도록 확장할 수도 있습니다.

더 나아가 현재 프로그램은 날짜 입력만 가능하도록 제한되어 있으나, 지역(시·도, 시·군·구)을 선택할 수 있는 기능을 추가한다면 사용자가 더욱 세밀한 조건으로 검색할 수 있게 됩니다. 이처럼 기능을 확장하고 UI 요소를 개선하는 과정을 통해 프로그램을 단순한 예제를 넘어 실제 활용 가능한 형태로 발전시킬 수 있으며, 동시에 안드로이드 앱 개발에 대한 이해와 응용력을 크게 높일 수 있을 것입니다.

연습문제

1. 공공데이터포털을 활용한 앱 개발 준비 단계를 설명하세요. 특히 회원가입, 마이페이지 접속, API 인증키 발급 및 관리 과정의 중요성을 중심으로 서술하세요.

2. 구조동물 조회 서비스 API를 호출하기 위해 필요한 주요 요청 파라미터(request parameter)와 이를 통해 얻을 수 있는 응답 정보(response parameter)를 예시를 들어 설명하세요.

3. 안드로이드에서 Retrofit 인터페이스와 NetworkModule이 어떤 역할을 수행하는지, 그리고 왜 이러한 구조를 사용하는 것이 효율적인지 설명하세요.

4. 제공된 예제에서 데이터 모델(ApiResponse 클래스)과 어댑터(AnimalAdapter 클래스)가 어떤 관계를 가지며, 두 요소가 앱 화면에 데이터를 표시하기 위해 어떻게 상호작용하는지 설명하세요.

5. 제공된 예제 앱은 날짜 기반 검색 기능만 구현되어 있습니다. 만약 지역(시 · 도, 시 · 군 · 구) 선택 기능을 추가하려 한다면, 어떤 API와 요청 파라미터를 활용해야 하며, UI 설계 측면에서 어떤 변경이 필요할지 설명하세요.

맺음말

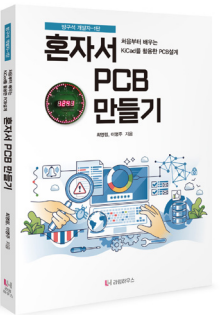
본 교재에서는 안드로이드 앱 개발의 기초를 단계적으로 다루었다. 개발 환경의 설치와 설정, 기본 레이아웃 구성, 이벤트 처리 방식, 어댑터를 활용한 데이터 표시, 그리고 공공데이터 API를 연동한 실습에 이르기까지 일련의 과정을 학습함으로써, 안드로이드 앱이 어떤 구조와 절차를 통해 구현되는지를 이해할 수 있도록 구성하였다.

앱 개발은 단순한 화면 구성 기술에 그치지 않고, 사용자 경험(UX)을 고려한 인터페이스 설계, 네트워크 및 데이터 관리, 예외 상황 처리 등 종합적인 요소가 유기적으로 결합되는 과정이다. 본 교재에서 다룬 내용은 기초 단계에 해당하지만, 이를 바탕으로 학습자는 보다 복잡하고 다양한 기능을 가진 애플리케이션으로 확장할 수 있는 토대를 마련할 수 있을 것이다.

앞으로는 본 교재에서 습득한 기본기를 기반으로, 다양한 오픈소스 라이브러리 활용, 클라우드 및 서버 연계, 고급 UI/UX 설계, 인공지능(AI)이나 사물인터넷(IoT)과 같은 최신 기술과의 결합 등으로 학습의 범위를 넓혀가는 것이 바람직하다.

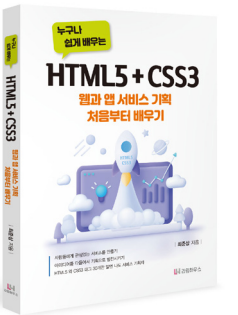
안드로이드 앱 개발은 지속적으로 진화하는 분야이므로, 꾸준한 학습과 실습을 통해 새로운 기술과 도구를 습득하는 자세가 필요하다. 본 교재가 향후 학습자에게 체계적인 이해와 실무 응용 능력 배양에 기초 자료로서 활용되기를 기대한다.

도서 안내



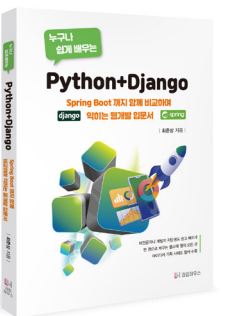
혼자서 PCB 만들기

지은이 : 최영림, 이영주
ISBN : 979-11-976826-5-0
정가 : 18,000
페이지 : 236



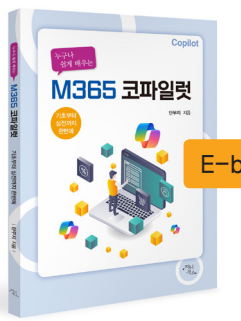
누구나 쉽게 배우는 HTML5+CSS3

지은이 : 최준삼
ISBN : 979-11-976826-4-3
정가 : 17,000
페이지 : 228



누구나 쉽게 배우는 Python+Django

지은이 : 최준삼
ISBN : 979-11-976826-9-8
정가 : 18,000
페이지 : 228



E-book 전용

누구나 쉽게 배우는 M365 코파일럿

지은이 : 단무지
ISBN : 979-11-94808-07-7
정가 : 14,000
페이지 : 184

GT 골든트리미디어

출판브랜드: 라임하우스, 지니북스, 이앤비플러스, DK로드북스

라임하우스 지니북스 E&B PLUS 이앤비플러스 DK Road BOOKS