



CHAP 2. 애플리케이션서 기본 구조

2025-03-03(Mon)

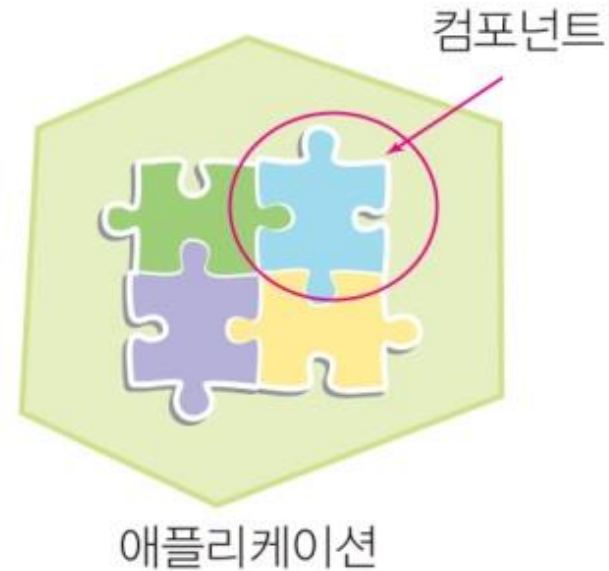


2장의 목표

- 첫 번째 앱을 만들어본다.



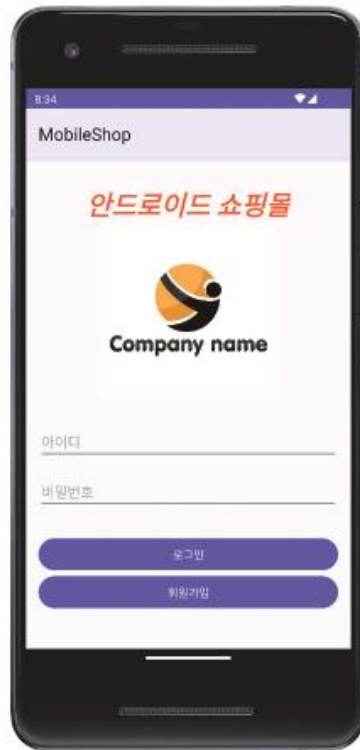
- 애플리케이션은 컴포넌트로 이루어진다.
 - 액티비티(activity)
 - 서비스(service)
 - 방송 수신자(broadcast receiver)
 - 콘텐츠 제공자(content provider)





액티비티

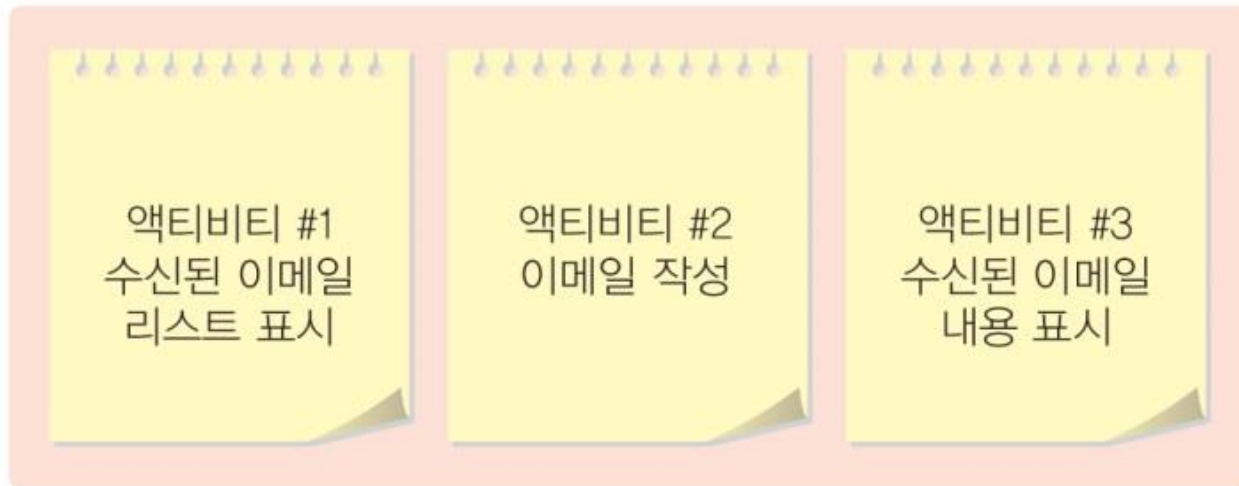
- 사용자 인터페이스 화면을 가지는 하나의 작업을 표시하는 컴포넌트





액티비티의 예

- 액티비티들이 모여서 애플리케이션이 된다.



이메일 애플리케이션

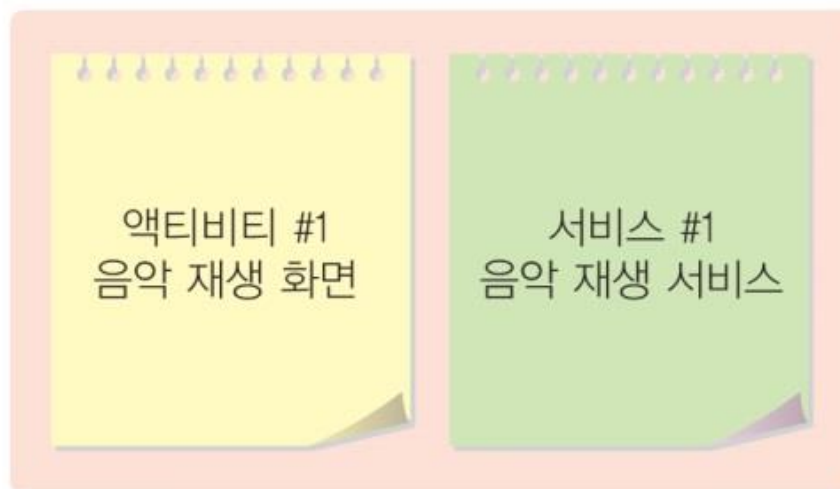


서비스

- 백그라운드에서 실행되는 컴포넌트로서 오랫동안 실행되는 작업이나 원격 프로세스를 위한 작업
- (예) 배경 음악을 연주하는 작업



서비스



미디어 플레이어 애플리케이션



방송 수신자

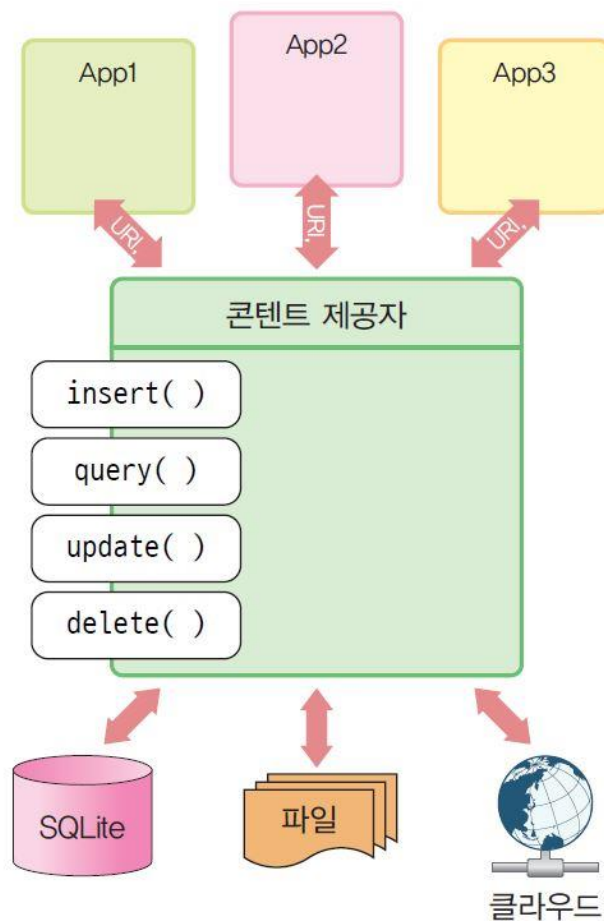
- 방송을 받고 반응하는 컴포넌트





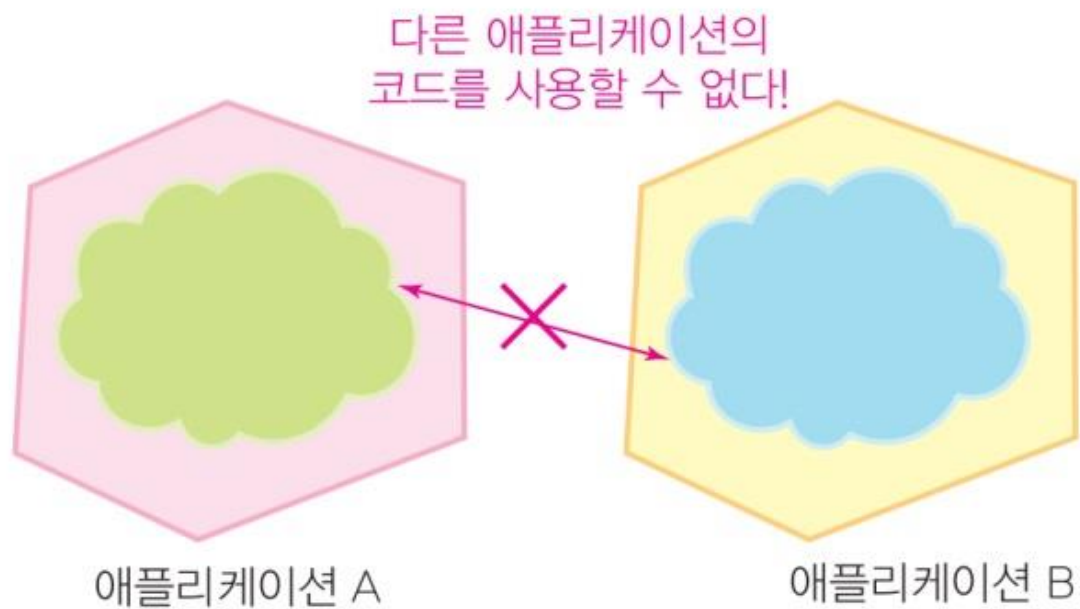
콘텐츠 제공자

- 데이터를 관리하고 다른 애플리케이션에게 제공하는 컴포넌트



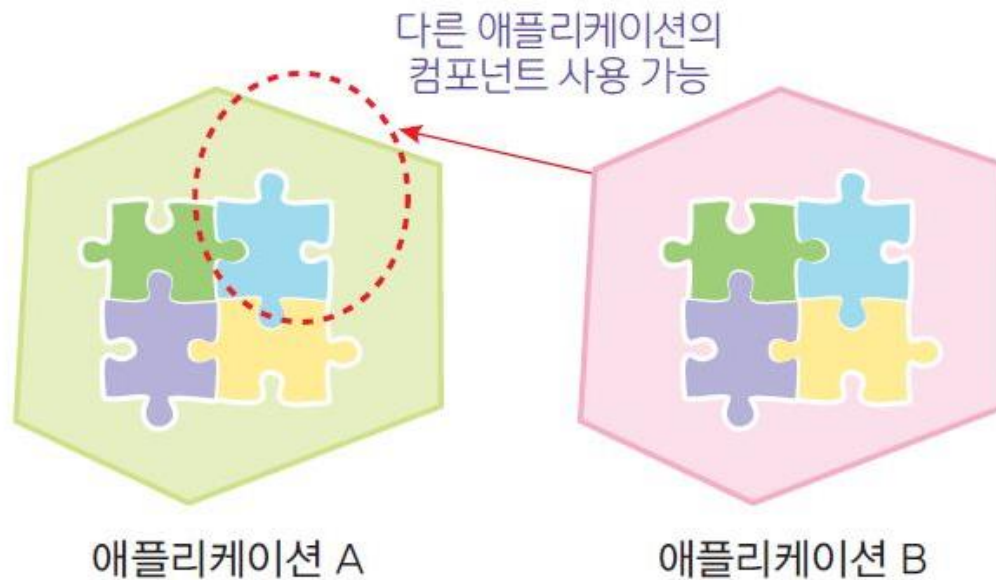


PC의 애플리케이션





안드로이드에서는 다른 컴포넌트를 사용할 수 있다





예제

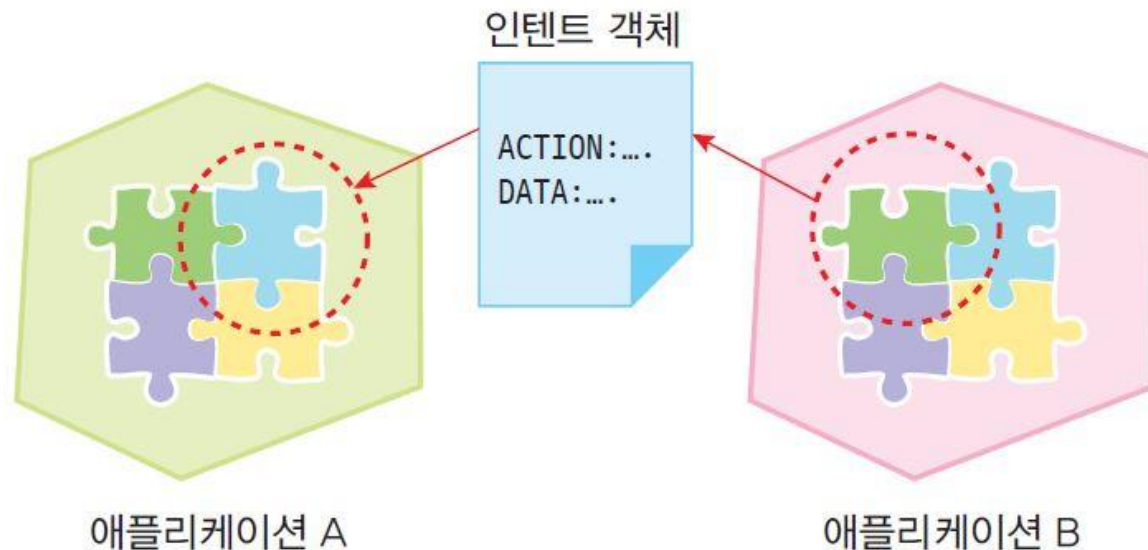
- 애플리케이션에서 사용자가 사진을 촬영하도록 하고 싶은 경우





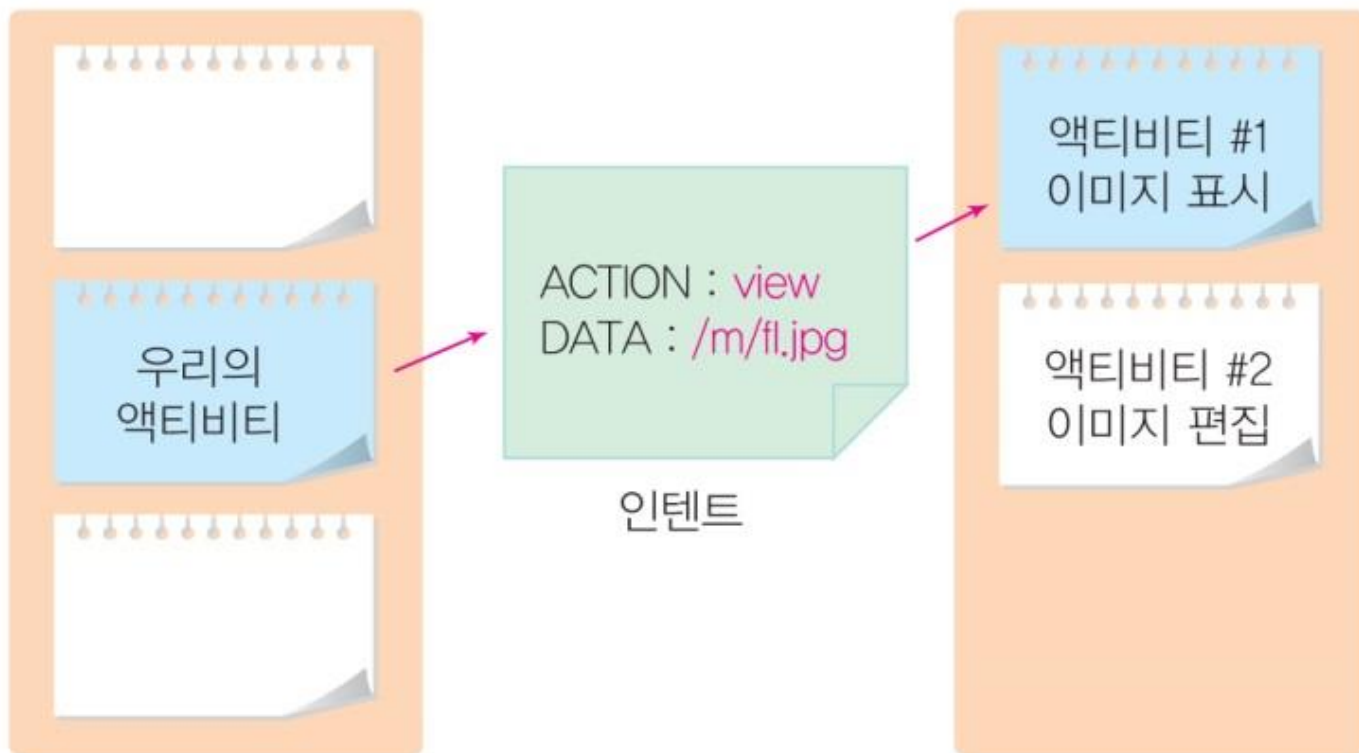
인텐트

- 애플리케이션의 의도를 적어서 안드로이드에 전달하면 안드로이드가 가장 적절한 컴포넌트를 찾아서 활성화하고 실행





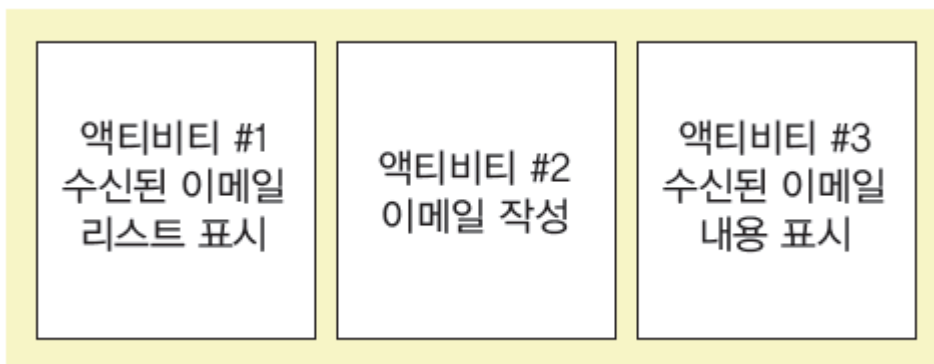
인텐트 사용의 예





앱을 개발할 때 4가지 컴포넌트를 전부 사용해야 할까?

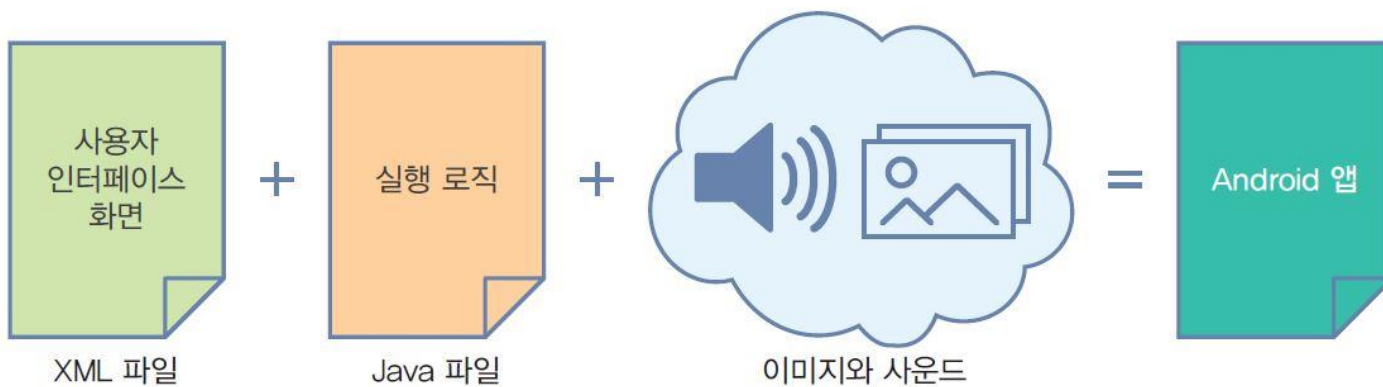
- 개발자가 앱을 개발할 때는 자신에게 필요한 컴포넌트가 무엇인지를 결정하여야 한다. 4가지 컴포넌트가 모두 있어야 하는 것은 결코 아니다.





애플리케이션의 구성

- 자바 파일은 앱의 로직을 나타내고, XML 파일은 앱의 사용자 인터페이스를 나타낸다. 여기에 사운드와 이미지 같은 리소드(자원)들이 추가된다.





일반적인 애플리케이션 작성 절차

- ① 사용자 인터페이스 작성(XML)
- ② 자바 코드 작성(JAVA)
- ③ 매니페스트 파일 작성(XML)





1. 사용자 인터페이스 작성

- 첫 번째 단계는 XML을 이용하여 사용자 인터페이스 화면을 디자인하는 단계





2. 자바 코드 작성

- 두 번째 단계는 자바를 이용하여서 코드를 작성하는 단계

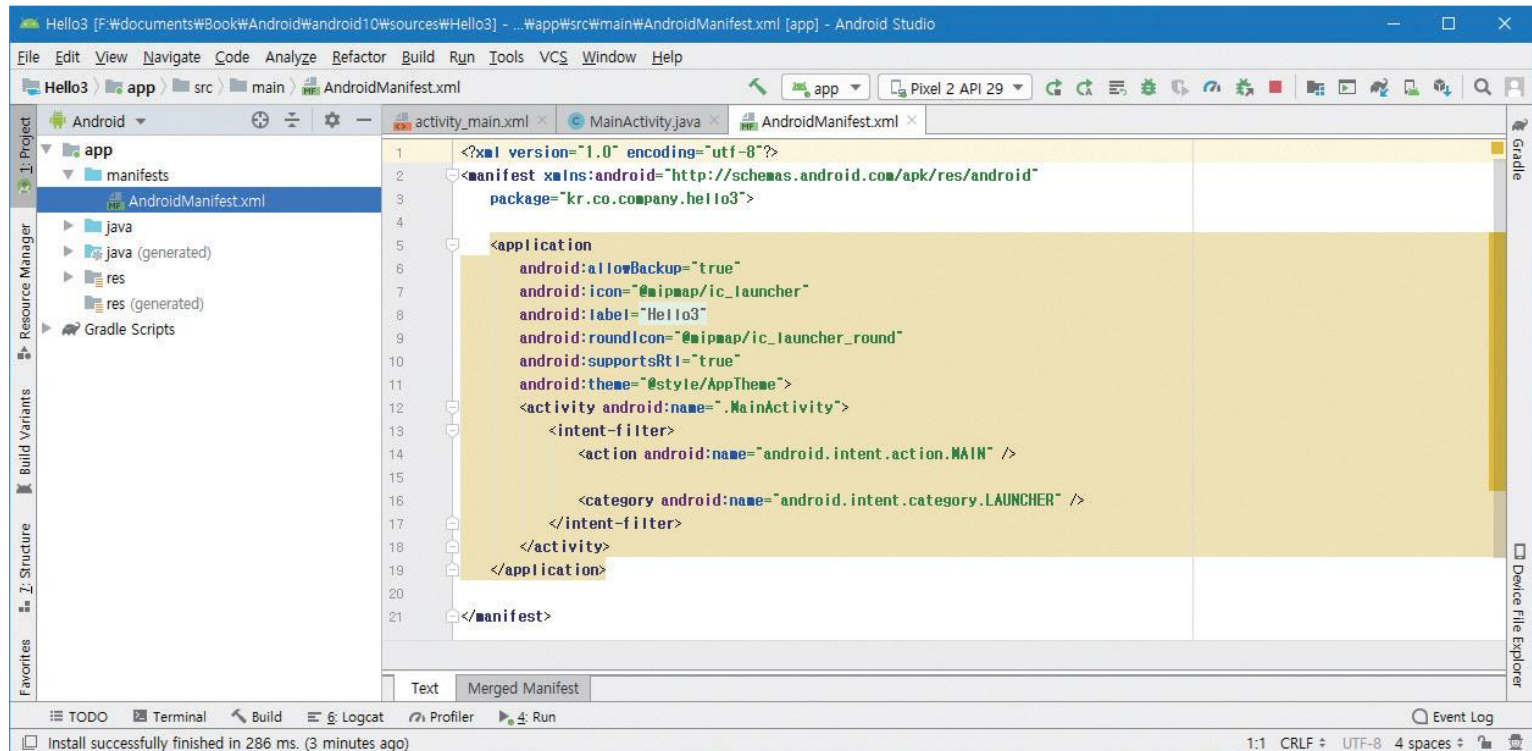
```
package kr.co.company.helloandroid;  
import android.app.Activity;  
import android.os.Bundle;  
public class HelloAndroid extends  
Activity  
{  
    @Override  
    public void onCreate(Bundle  
savedInstanceState)  
    {  
        super.  
onCreate(savedInstanceState);  
        setContentView(R.layout.  
activity_main);  
    }  
}
```





3. 매니페스트 파일 작성

- 애플리케이션을 구성하고 있는 컴포넌트를 기술하고 실행 시에 필요한 권한을 지정한다.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="kr.co.company.hello3">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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



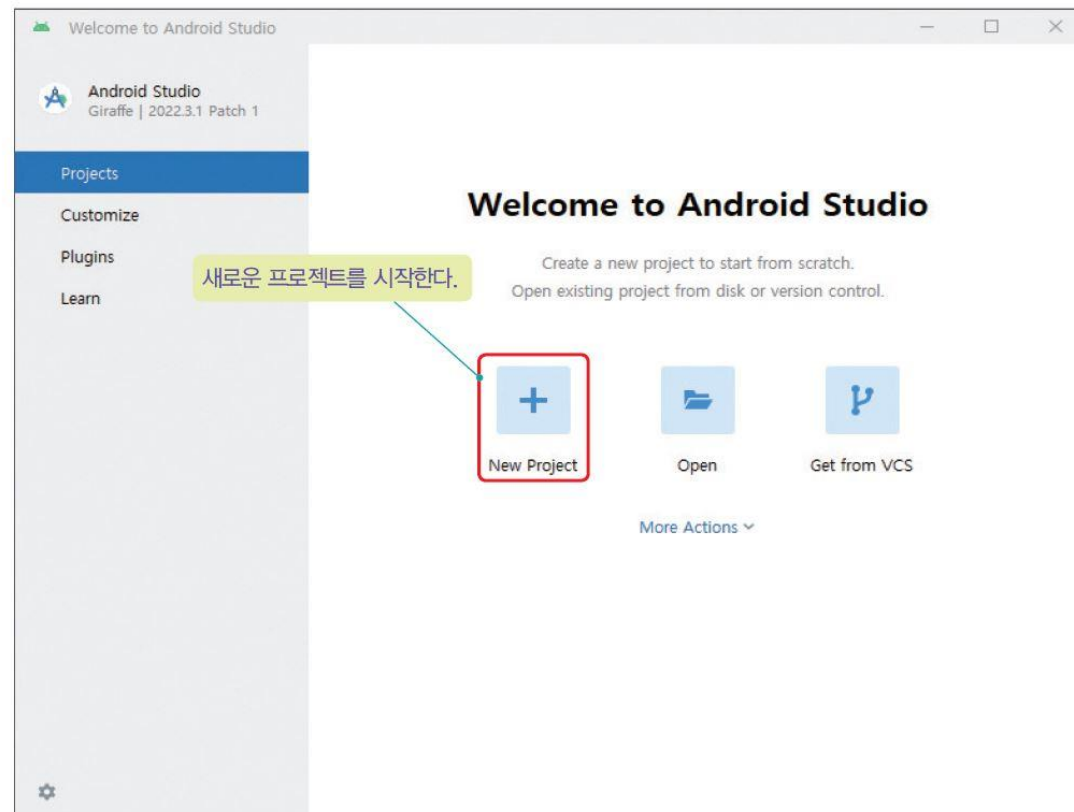
자바 vs 코틀린

- 최근에 안드로이드 앱을 개발할 때 코틀린(Kotlin)과 자바(Java) 중에서 하나를 선택하여야 한다.
- 언어 선택은 개발자의 선호도와 프로젝트 요구사항에 따라 다를 수 있다. 최근 구글에서는 코틀린을 더 권장하고 있다. 코틀린은 문법이 간결하고 읽기 쉬운 언어로, 코드 작성과 유지보수가 더 쉽다.
- 코틀린은 기존의 자바 코드와 원활하게 통합되므로, 이미 자바로 작성된 코드와 함께 사용하기 쉽다.
- 하지만 코틀린은 자바와는 상당히 문법이 달라서 아직도 상당수의 개발자는 전통적인 개발 언어인 자바를 선호하고 있다. 이 책에서는 자바를 사용한다.



실습 예제: 첫 번째 앱 만들기

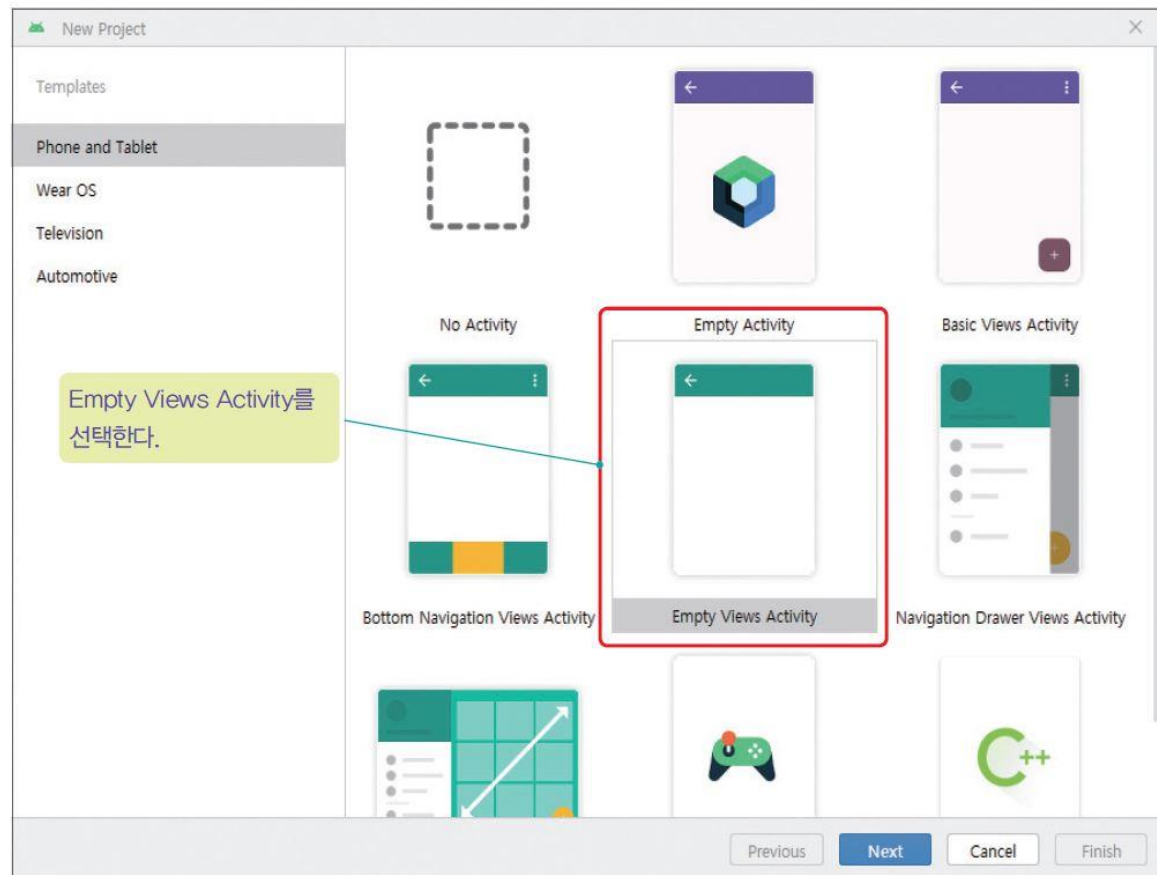
- (1) 윈도우의 [시작] 메뉴에서 안드로이드 스튜디오를 찾아서 실행한다.
- (2) 시작 화면에서 [New Project]를 선택한다.





실습 예제: 첫 번째 앱 만들기

(3) 애플리케이션의 유형을 선택한다. [Phone and Tablet] 탭의 [Empty View Activity]를 선택한다.





실습 예제: 첫 번째 앱 만들기

(4) 애플리케이션 이름과 회사 도메인, 프로젝트 위치, 사용 언어를 다음과 같이 입력한다.

The screenshot shows the 'New Project' dialog in Android Studio. The dialog has a title bar 'New Project' and a close button. The main content area is titled 'Empty Views Activity' and has a subtitle 'Creates a new empty activity'. There are five input fields, each with a red border and a corresponding annotation in a yellow box:

- Name:** 'Hello' (Annotation: ① "Hello"를 입력한다.)
- Package name:** 'kr.co.example.hello' (Annotation: ② 자바 패키지 이름을 입력한다.)
- Save location:** 'C:\Users\kim\AndroidStudioProjects\Hello' (Annotation: ③ 프로젝트가 생성되는 폴더를 지정한다.)
- Language:** 'Java' (Annotation: ④ 자바를 선택한다.)
- Minimum SDK:** 'API 24 ("Nougat"; Android 7.0)' (Annotation: ⑤ 앱이 실행되는 최하위 버전이다. 95% 장치에서 실행이 가능하다고 알려주고 있다.)

Below the input fields, there is a blue information icon and text: 'Your app will run on approximately 95.4% of devices.' with a link 'Help me choose'. At the bottom, there is a 'Build configuration language' dropdown menu set to 'Kotlin DSL (build.gradle.kts) [Recommended]'. At the very bottom, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.



애플리케이션의 구성

The screenshot displays the Android Studio interface with three main components highlighted by pink boxes:

- 프로젝트 뷰 (Project View):** Located on the left, it shows the project structure. The `com.example.hello` package is selected, containing `MainActivity` and test variants.
- 자바 소스 (Java Source):** The central editor shows the `MainActivity.java` file. The code defines the package, imports, and the `onCreate` method that sets the content view to `R.layout.activity_main`.
- 컴파일시의 메시지를 표시한다. (Show messages during compilation):** The bottom panel shows the `Build` output, indicating a successful build with 28 actionable tasks.

```
1 package com.example.hello;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
```

Build Output:

```
Build: finished At 2021-11-24 오후 7:25 30 sec, 592 ms
BUILD SUCCESSFUL in 29s
28 actionable tasks: 27 executed, 1 up-to-date
Build Analyzer results available
```




패키지 폴더의 설명

폴더 또는 파일	설명
manifest	XML 파일로 앱의 전반적인 정보, 즉 앱의 이름이나 컴포넌트 구성과 같은 정보를 가지고 있다.
java	자바 소스 파일들이 들어있는 폴더이다. 폴더 안의 <code>kr.co.example.hello</code> 는 패키지의 이름이다.
res	각종 리소스(자원)들이 저장되는 폴더이다. <code>drawable</code> 에는 해상도 별로 아이콘 파일들이 저장된다. <code>layout</code> 에는 화면의 구성을 정의한다. <code>values</code> 에는 문자열과 같은 리소스가 저장된다. <code>menu</code> 에는 메뉴 리소스들이 저장되어 있다.
Gradle Scripts	그레이들(Gradle)은 빌드 시에 필요한 스크립트이다.
MainActivity.java	메인 액티비티를 나타내는 자바 소스 파일
activity_main.xml	메인 액티비티의 화면을 나타내는 XML 파일

● 표 2-1
프로젝트 뷰의
중요한 폴더와
파일들



자동 생성된 소스 관찰

The screenshot shows the Android Studio interface with the Project view on the left. The project structure is as follows:

- app** (Module)
 - manifests**
 - AndroidManifest.xml** (XML file)
 - java** (Folder)
 - kr.co.example.hello** (Package)
 - MainActivity** (Java class)
 - kr.co.example.hello (androidTest)** (Test package)
 - kr.co.example.hello (test)** (Test package)
 - java (generated)** (Folder)
 - res** (Folder)
 - drawable** (Folder)
 - layout** (Folder)
 - activity_main.xml** (XML file)
 - mipmap** (Folder)
 - values** (Folder)
 - xml** (Folder)
 - res (generated)** (Folder)
- Gradle Scripts** (Folder)
 - build.gradle.kts (Project: Hello)**
 - build.gradle.kts (Module :app)**
 - proguard-rules.pro (ProGuard Rules for ":app")**
 - gradle.properties (Project Properties)**
 - gradle-wrapper.properties (Gradle Version)**
 - local.properties (SDK Location)**
 - settings.gradle.kts (Project Settings)**

Annotations and explanations:

- app**: 앱이 시작된다.
- AndroidManifest.xml**: XML 파일로 앱의 전반적인 정보, 즉 앱의 이름이나 컴포넌트 구성과 같은 정보를 가지고 있다.
- java**: 자바 소스 파일들이 들어있는 폴더이다.
- kr.co.example.hello**: 폴더 안의 kr.co.example.hello는 패키지 이름이다.
- MainActivity**: 메인 액티비티를 나타내는 자바 소스 파일
- res**: 각종 리소스(자원)들이 저장되는 폴더이다. drawable에는 해상도 별로 아이콘 파일들이 저장된다. layout에는 화면의 구성을 정의한다. values에는 문자열과 같은 리소스가 저장된다. menu에는 메뉴 리소스들이 저장되어 있다.
- activity_main.xml**: 메인 액티비티의 화면을 나타내는 XML 파일
- Gradle Scripts**: 그레이들(Gradle)은 빌드 시에 필요한 스크립트이다.



일단 2개의 파일이 중요

- *MainActivity.java* 파일은 앱의 메인 액티비티(Activity)를 정의하는 파일이다. 메인 액티비티는 앱이 시작될 때 가장 먼저 보여지는 화면이며, 사용자 인터페이스와 액티비티의 동작을 관리한다.
- *activity_main.xml* 파일은 MainActivity.java에서 정의한 액티비티의 사용자 인터페이스를 설계하고 레이아웃을 정의하는 파일이다.



MainActivity.java

```
package kr.co.company.hello;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context=".MainActivity">

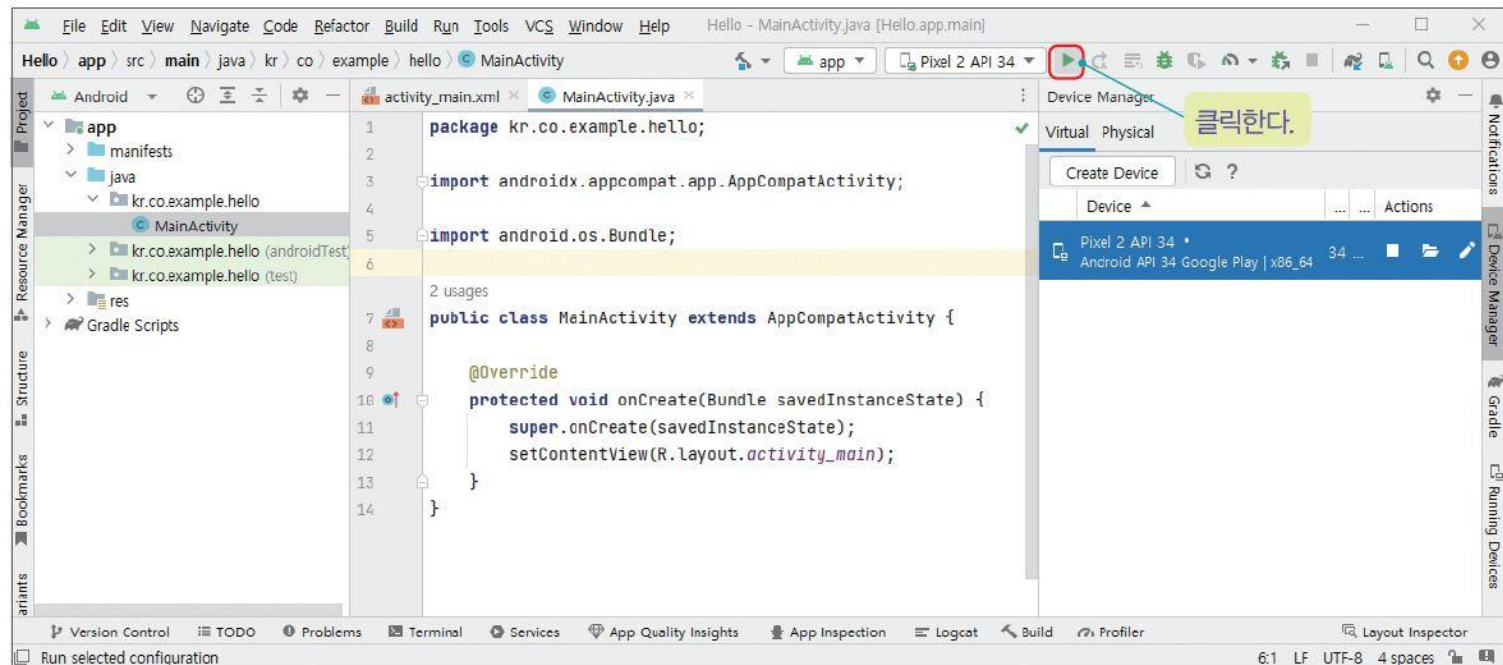
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



앱 실행하기

- AVD에서 앱을 실행하기 위하여 실행 버튼을 누른다.





앱 실행하기

The screenshot shows the Android Studio IDE with the following components:

- Project View:** Shows the project structure with 'app' as the main module, containing 'manifests', 'java', 'res', and 'Gradle Scripts'.
- Code Editor:** Displays the `MainActivity.java` file with the following code:

```
1 package kr.co.example.hello;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
```
- Running Devices:** A virtual Pixel 2 API 34 device is running. The screen displays 'Hello World'.
- Callout Box:** A yellow box with Korean text points to the device screen: "자동 생성된 앱이다. 화면의 중앙에 'Hello World!'를 출력한다." (This is an auto-generated app. It outputs 'Hello World!' in the center of the screen.)
- Status Bar:** At the bottom, it says "Install successfully finished in 1 s 416 ms. (moments ago)".



자바 소스 분석

MainActivity.java

```
package kr.co.company.hello;
```

← 패키지 지정 문장

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

```
import android.os.Bundle;
```

← 필요한 클래스를 포함시키는 import 문장들

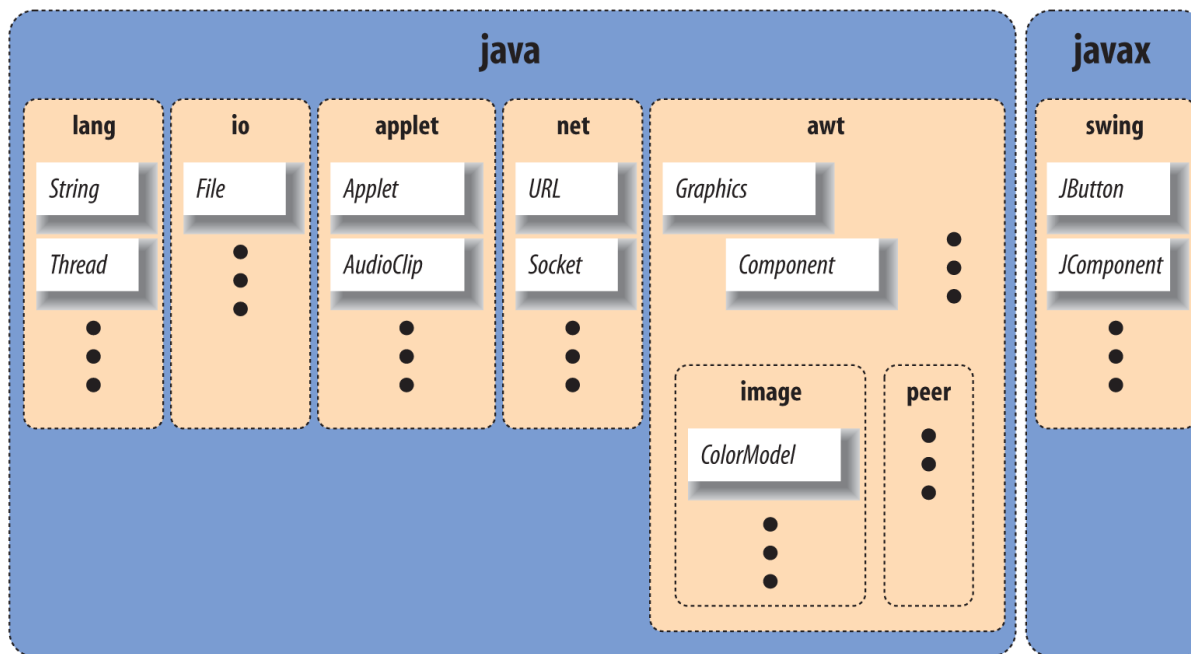
```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

← MainActivity
클래스 정의



package kr.co.company.hello;

- 패키지(**package**)는 클래스들을 보관하는 컨테이너
- 일반적으로 인터넷의 도메인 이름을 역순으로 사용





import

androidx.appcompat.app.AppCompatActivity...

- import 문장은 외부에서 패키지나 클래스를 포함
- 앞에 androidx가 붙은 패키지는 JetPack에 속하는 클래스로서 호환성을 위하여 최근에 사용이 권장되는 패키지이다.



public class MainActivity extends AppCompatActivity { ... }

- 클래스의 정의
- **Activity**로부터 상속받았으므로 액티비티가 된다.
- 액티비티는 안드로이드에서 애플리케이션을 구성하는 4가지의 컴포넌트 중의 하나이다.



액티비티는 화면을 통하여 사용자와 상호작용하는 활동을 의미한다.



@Override

- 어노테이션의 하나
- 어노테이션은 컴파일러에게 추가적인 정보를 주는 것
- **@Override**은 메소드가 부모 클래스의 메소드를 재정의(오버라이드)하였다는 것을 나타낸다.



public void onCreate() { ... }

- onCreate() 메소드는 액티비티가 생성되는 순간에 딱 한번 호출
- 모든 초기화와 사용자 인터페이스 설정이 여기에 들어간다.



`super.onCreate(savedInstanceState);`

- 위의 문장은 부모 클래스인 AppCompatActivity 클래스의 onCreate()를 호출하는 문장



setContentView(R.layout.activity_main);

- setContentView()라는 함수는 액티비티의 화면을 설정하는 함수
- R.layout.activity_main은 activity_main.xml 파일을 나타낸다.



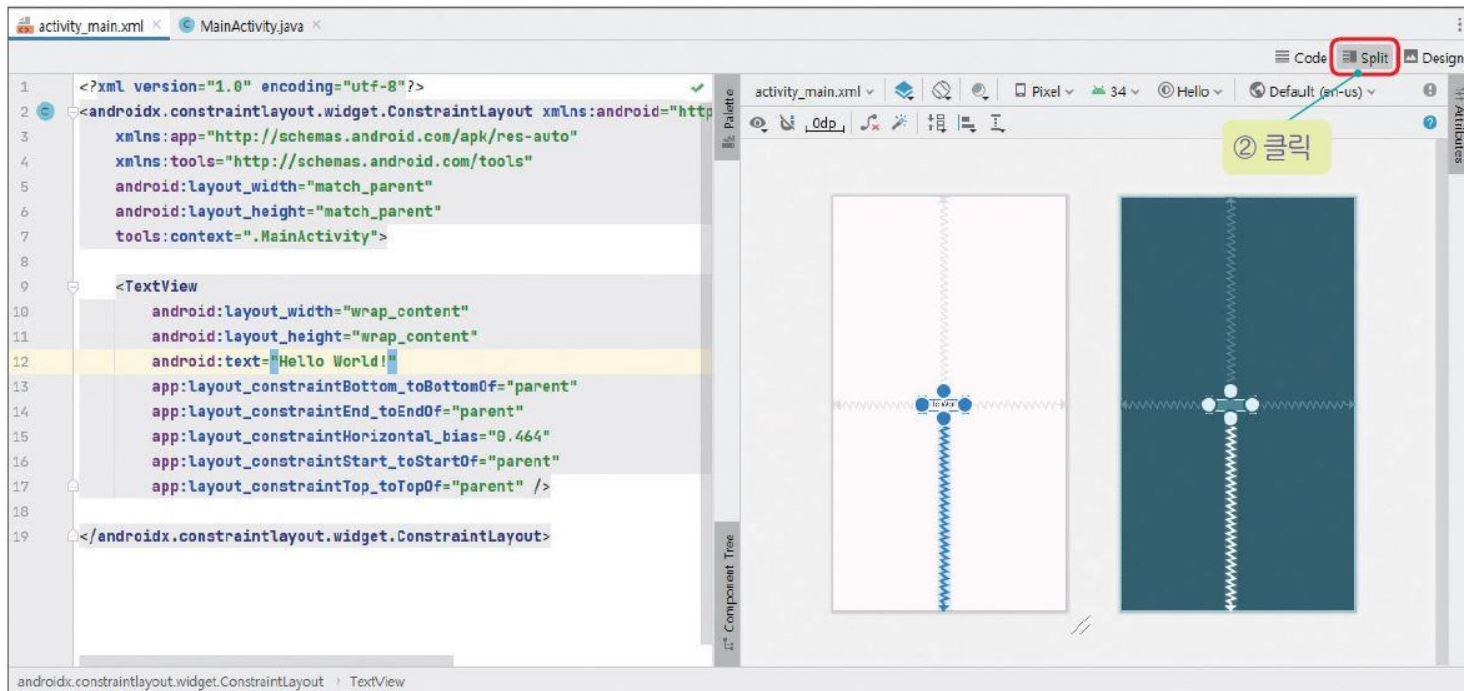
안드로이드 애플리케이션의 실행이 시작되는 곳

- 안드로이드에는 `main()`이 없음.
- 액티비티별로 실행된다.
- 액티비티 중에서는 `onCreate()` 메소드가 가장 먼저 실행된다.



activity_main.xml

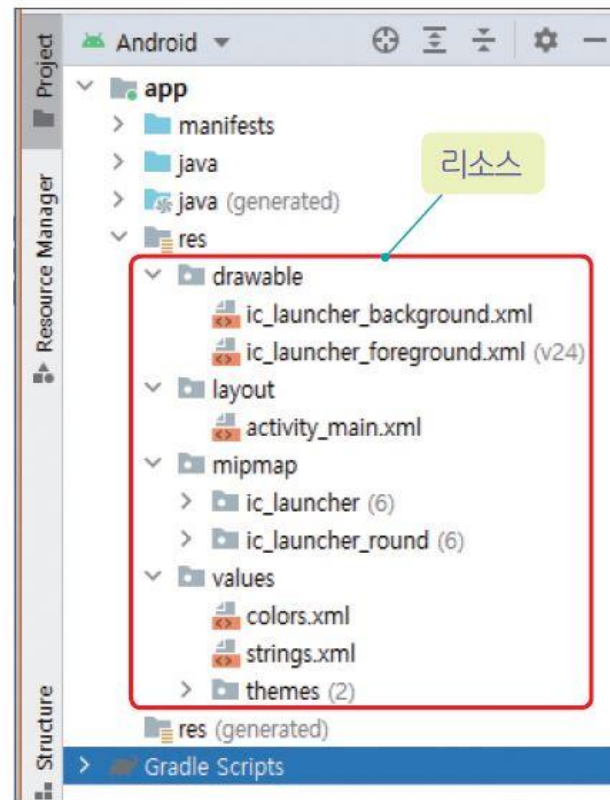
- 화면의 패키지 탐색기에서 **res** 폴더 아래에 보면 **layout** 폴더가 있고 그 안에 **activity_main.xml**이라는 파일이 있다.





리소스

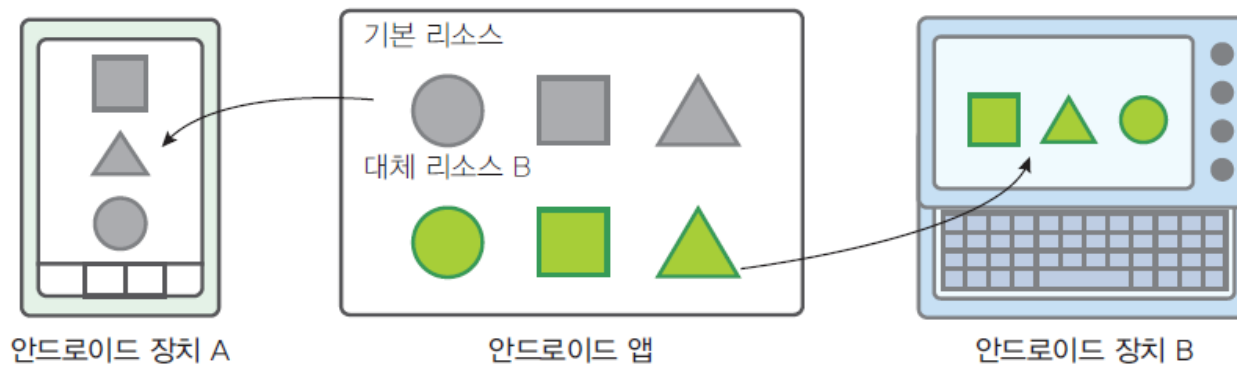
- 안드로이드에서 레이아웃, 이미지, 문자열 등은 리소스로 취급된다.
- 리소드들은 모두 XML을 이용하여 정의된다.





코드와 리소스의 분리

- 코드는 프로그래머가 작성하고 리소스는 디자이너가 작성한다.





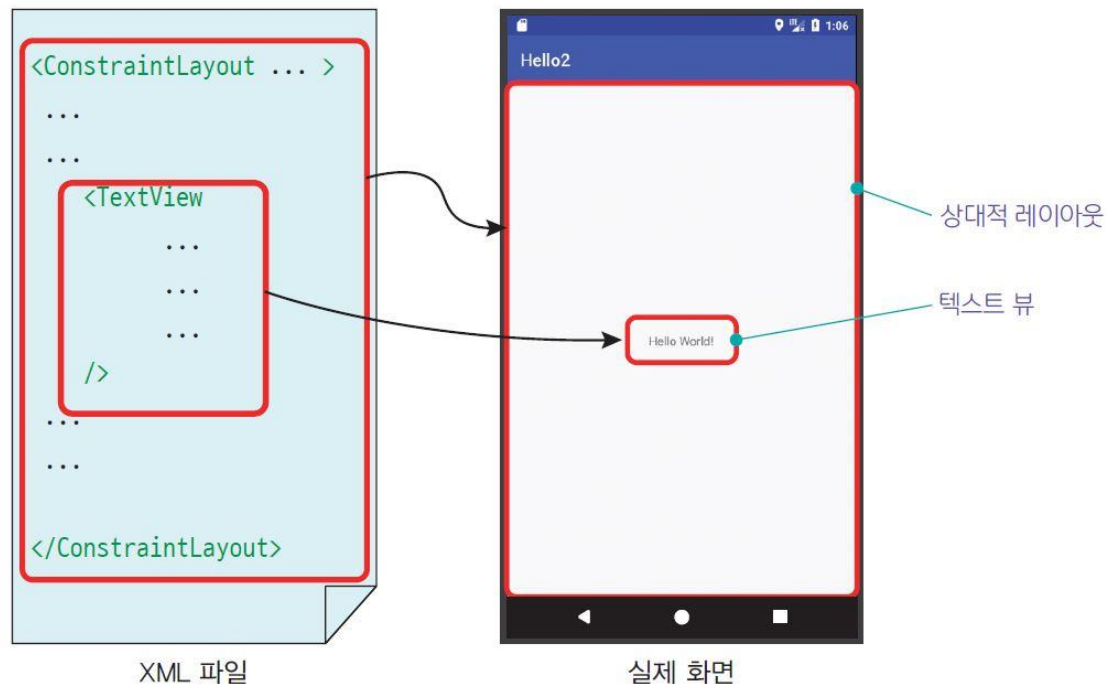
XML을 이용하여서 사용자 인터페이스 기술

- 사용자 인터페이스 작성 방법
 - 코드를 사용하는 방법(기존의 자바)
 - XML을 사용하는 방법(안드로이드 선호 방법)
- 안드로이드에서는 **UI** 화면의 구성을 XML을 이용하여서 선언적으로 나타내는 방법을 선호
 - 애플리케이션의 외관과 애플리케이션의 로직을 서로 분리
 - 빠르게 **UI**를 구축



레이아웃

- 레이아웃(layout)이란 화면을 어떻게 구성할 것이냐 하는 것이다. 즉 어떠한 위젯들을 선택하고 어떻게 화면에 배치할 것인지를 레이아웃에서 결정한다.





- 시작 태그로 시작되어 종료 태그로 끝나는 논리적인 구성 요소를 **요소(element)**
- `<Greeting>Hello, world.</Greeting>`가 요소
- 속성(**attribute**)은 요소의 속성으로서 “이름/값”의 쌍으로 구성
- ``에서는 img 요소는 src와 alt라는 2개의 속성을 가진다.



XML 속성

속성	의미
<code>xmlns:android</code>	XML 이름공간의 선언으로 안드로이드 이름공간에 정의된 속성들을 참조하려고 한다는 것을 암시한다. XML 파일에서 항상 최외곽 태그는 이 속성을 정의하여야 한다.
<code>android:id</code>	TextView 요소에 유일한 아이디를 할당한다. 이 아이디를 이용하여서 소스 코드에서 이 텍스트 뷰를 참조할 수 있다.
<code>android:layout_width</code>	화면에서 얼마나 폭을 차지할 것인지를 정의한다. "match_parent"는 부모 화면의 폭을 다 차지하는 것을 의미한다.
<code>android:layout_height</code>	화면에서 높이를 얼마나 차지할 것인지를 정의한다. "wrap_content"는 콘텐츠를 표시할 정도만 차지하는 것을 의미한다.
<code>android:text</code>	화면에 표시하는 텍스트를 설정한다. 이 속성은 예제와 같이 하드코딩될 수도 있고 아니면 문자열 리소스의 개념을 사용할 수도 있다.



문자열 리소스

- 문자열도 XML로 기술하는 것이 바람직하다.
- 영어 버전, 한국어 버전, ...

strings.xml

```
<resources>
<string name="app_name">Hello</string>
<string name="message">Hello World!</string>
</resources>
```

activity_main.xml

```
...
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/message"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```




코드에서 리소스를 참조하는 방법

- 우리는 액티비티의 화면을 지정할 때, 레이아웃 리소스의 **ID(식별자)**를 사용한다.

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

← 레이아웃 리소스의 식별자



R.Java 파일

- 안드로이드 스튜디오는 **res** 폴더 아래의 리소스들을 분석하여서 각 리소스마다 겹치지 않는 식별자를 하나씩 부여한다.
- 이들 식별자는 **R.java**라고 하는 하나의 자바 파일에 모여 있게 된다.

```
...  
public class MainActivity extends Activity {  
  
    @Override  
    public void onCreate(...) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

MainActivity.java

```
public final class R {  
    ...  
    public static final class layout {  
        public static final int  
            activity_main=0x7f030000;  
    }  
    ...  
}
```

R.java

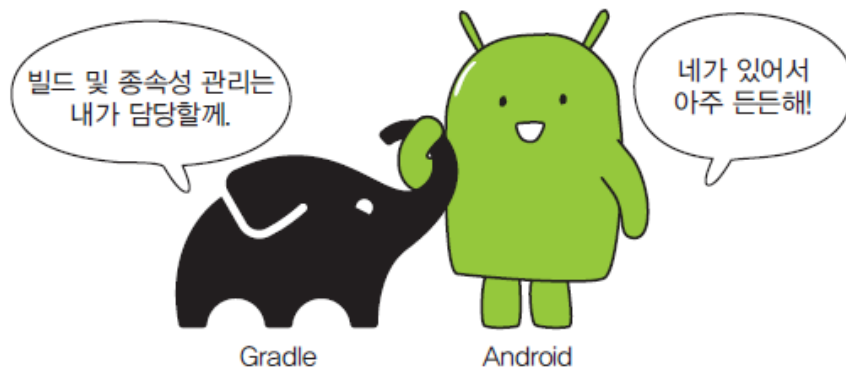
```
<?xml version="1.0" encoding="utf-8"?>  
<TextView xmlns:android="http://schemas  
    .android.com/apk/res/android"  
    android:id="@+id/textview"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Hello, world!" />
```

activity_main.xml



그레이들

- 그레이들(Gradle)은 안드로이드 앱의 빌드(build) 도구이다.
- 앱을 빌드하는데 필요한 라이브러리 버전도 자동으로 파악해서 필요하다면 다운로드한다.
- `build.gradle(Project)`는 전체 프로젝트에 대한 빌드 설정
- `build.gradle(Module)`에는 앱을 빌드하는데 필요한 중요한 설정 사항들이 많이 저장되어 있다.





그레이드

build.gradle(Module)

```
plugins {  
    id("com.android.application")  
}
```

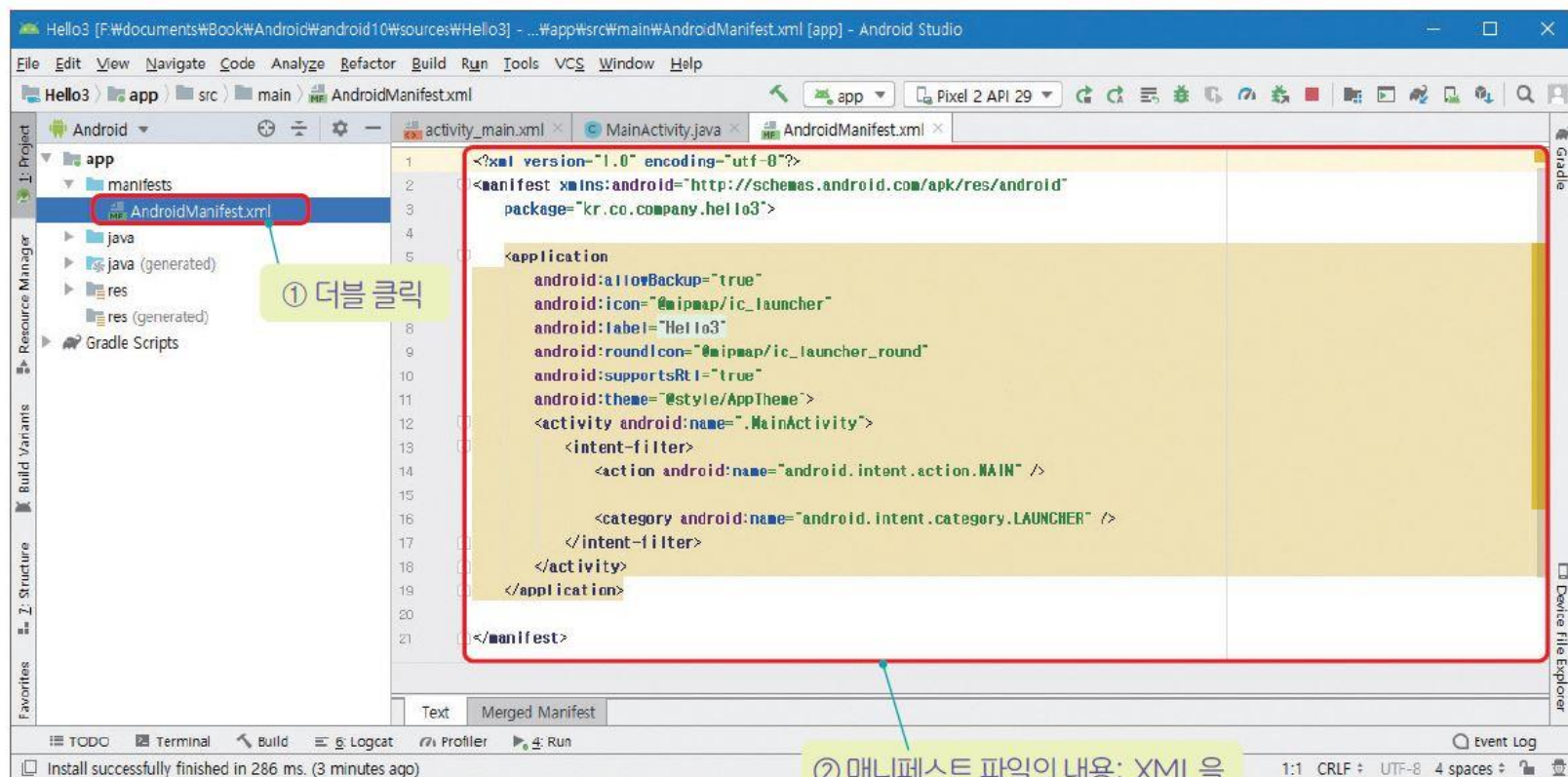
```
android {  
    namespace = "kr.co.example.hello"  
    compileSdk = 34 ← 컴파일러가 사용하는 SDK 버전
```

```
defaultConfig {  
    applicationId = "kr.co.example.hello"  
    minSdk = 24 ← 이 앱을 설치할 수 있는 장치의 최소 SDK 버전  
    targetSdk = 33 ← 목표로 하는 타겟 장치의 SDK 버전  
    versionCode = 1  
    versionName = "1.0"  
  
    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
}
```



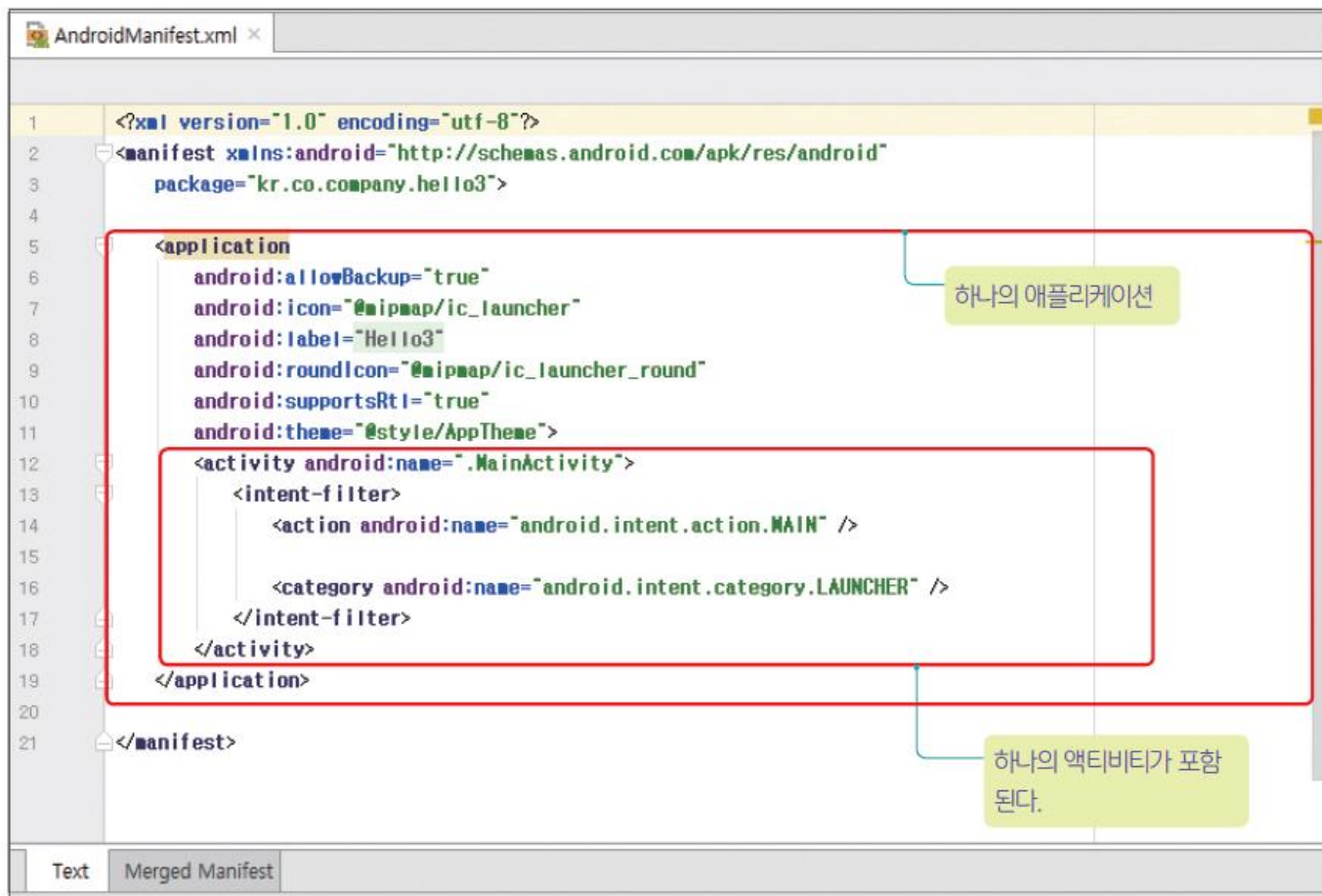
매니페스트 파일

- 매니페스트 파일은 애플리케이션에 적재된 모든 컴포넌트에 대하여 기술하는 파일이다.





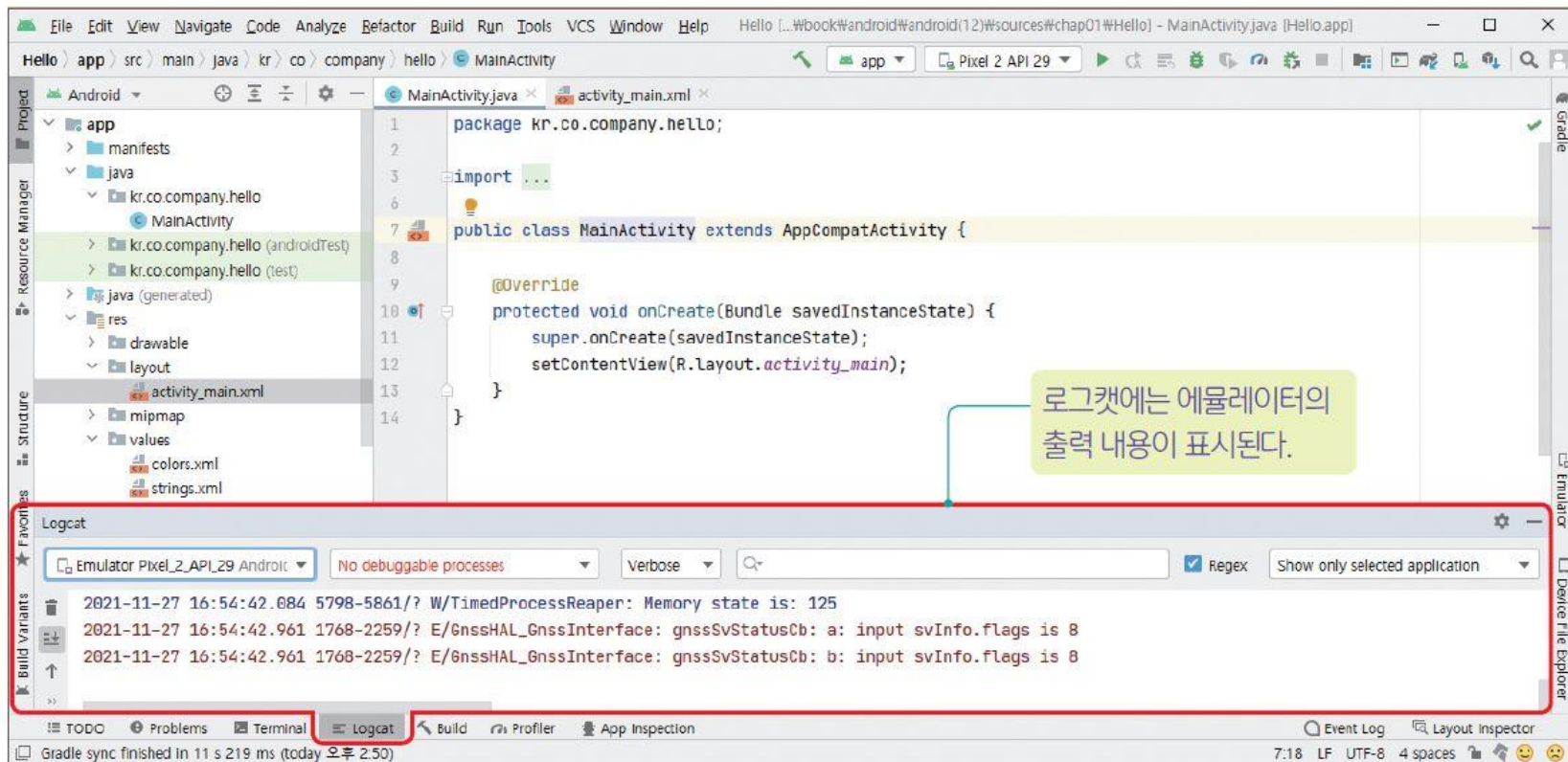
매니페스트 파일





에뮬레이터 로그캐

- 안드로이드 스튜디오 맨아래의 탭 중에서 [Logcat] 탭을 선택하면 실행되는 앱이 출력하는 메시지들을 볼 수 있다.





로그캣에 출력하려면

- 만약 개발자가 애플리케이션이 실행되는 도중에 어떤 값을 출력해보려면 로그캣을 이용할 수 있다.
- 다음과 같은 문장을 소스 코드 안에 포함시키면 에뮬레이터가 로그캣 창에 출력한다.

```
Log.v("MY_TAG", "App Start!!");
```

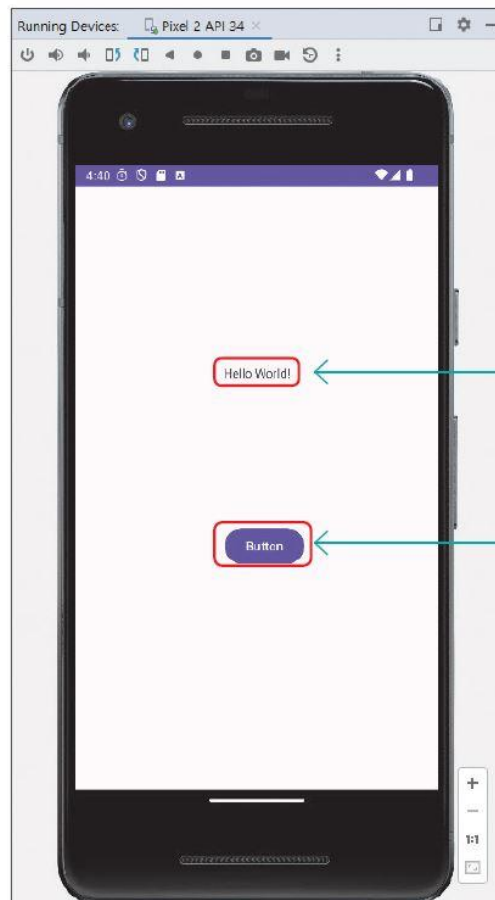
태그 이름, 로그캣 필터링에 사용한다.

메시지



예제: 비주얼 도구 사용해보기 |

- 다음과 같은 화면을 구성해보자.



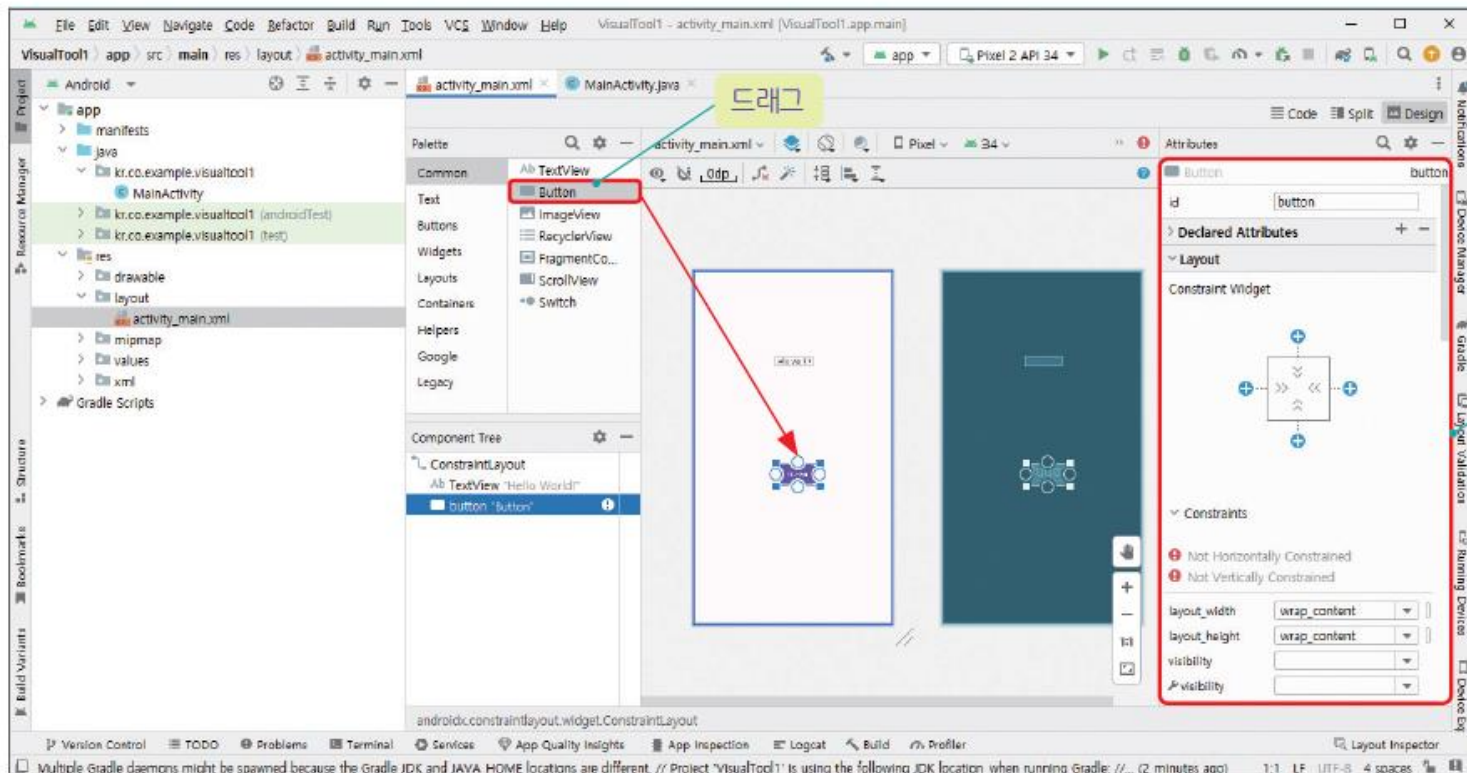
첫 번째 위젯인 TextView이다.

버튼을 눌러보자. 눌러지지만 아직까지 아무런 반응은 없는 데 이것은 당연하다. 반응을 주려면 버튼의 이벤트를 자바 언어로 처리하여야 한다.



Lab: 비주얼 도구 사용해보기 |

- 마우스로 위젯을 드래그하여 화면을 구성해보자.

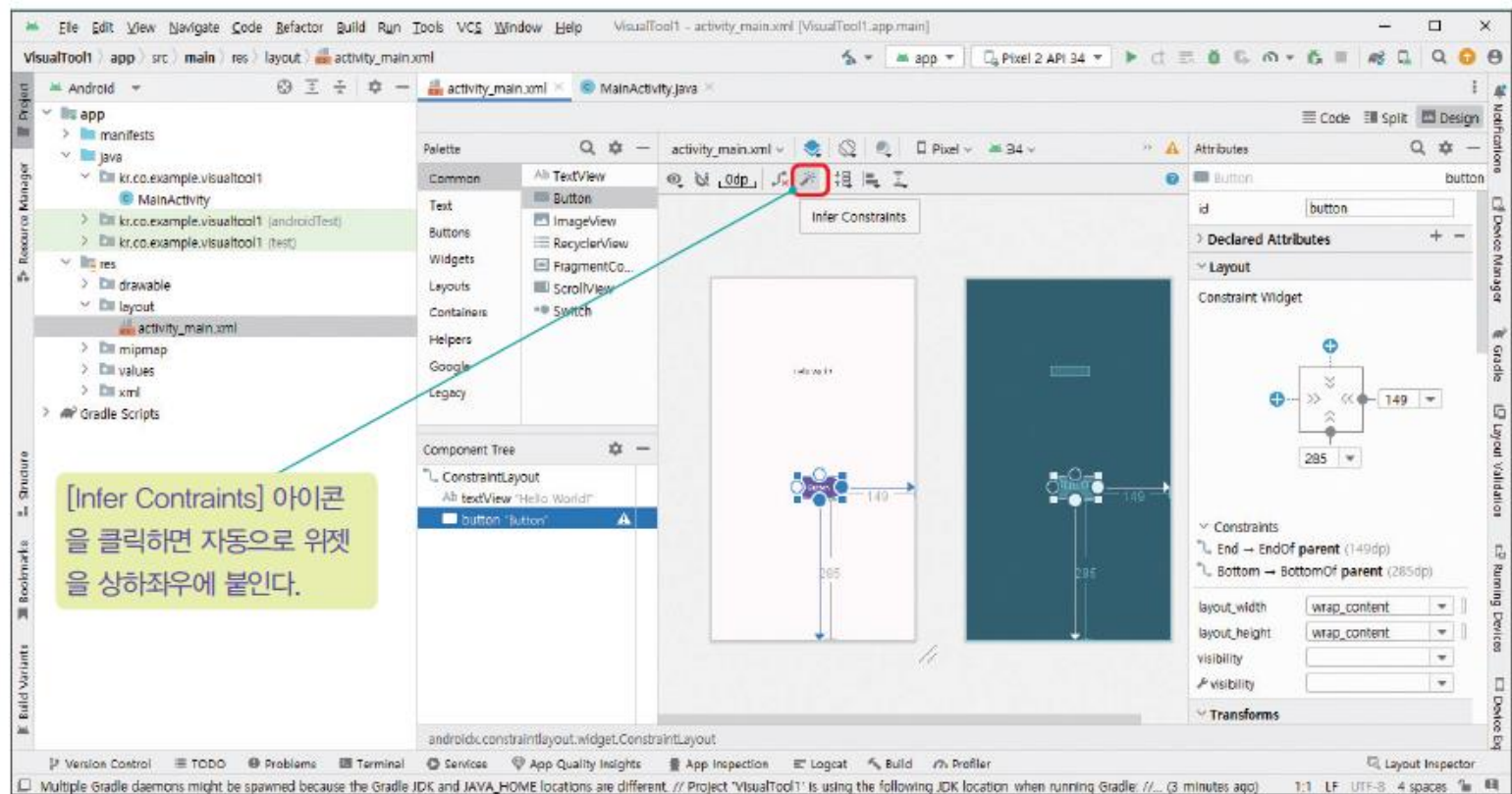


선택된 위젯의 속성이 나타난다. 우리가 변경할 수 있다.



Lab: 구속 조건 자동 생성

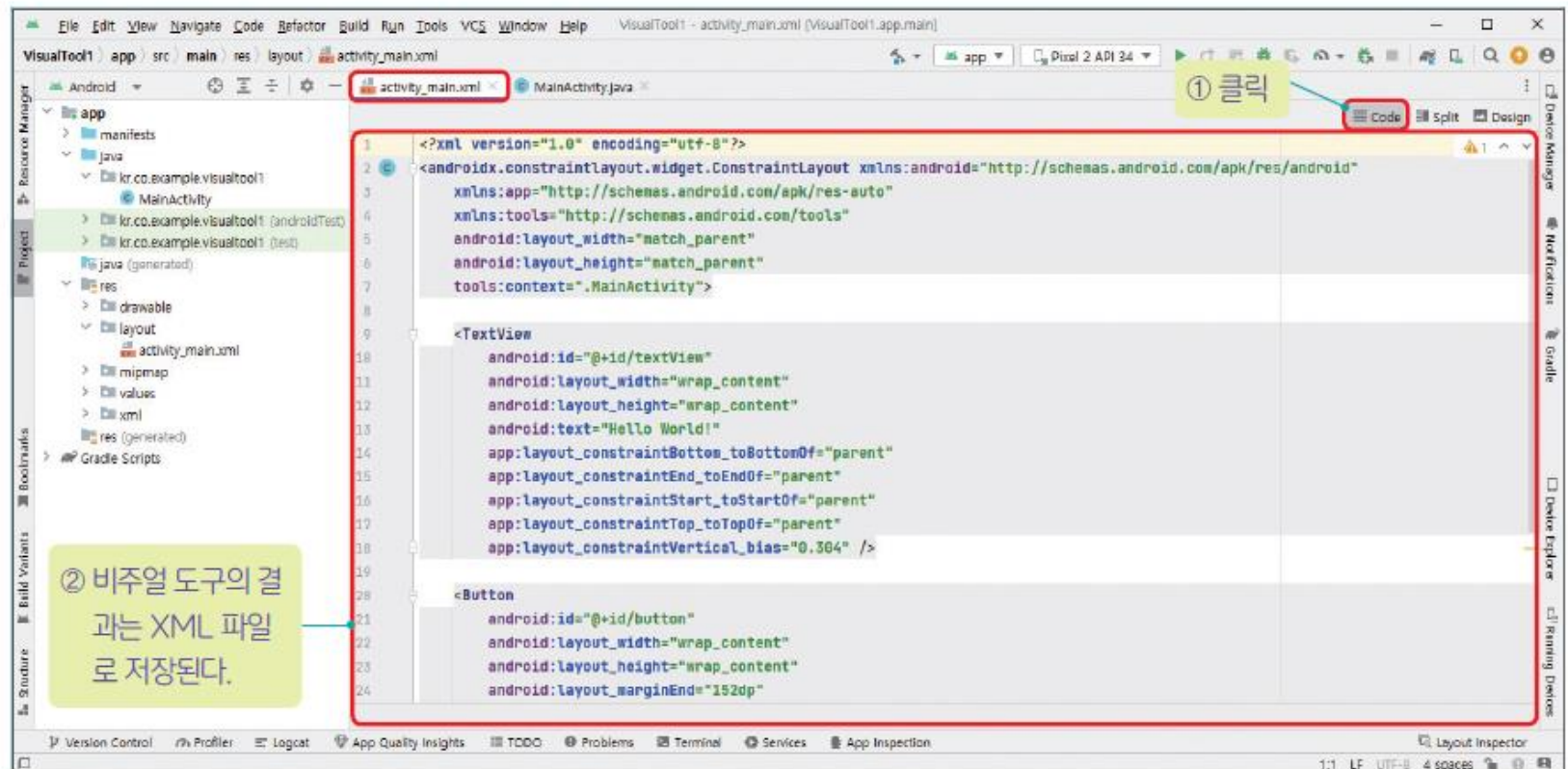
- 버튼을 선택한 상태에서 [Infer Constraints] 아이콘을 클릭한다.

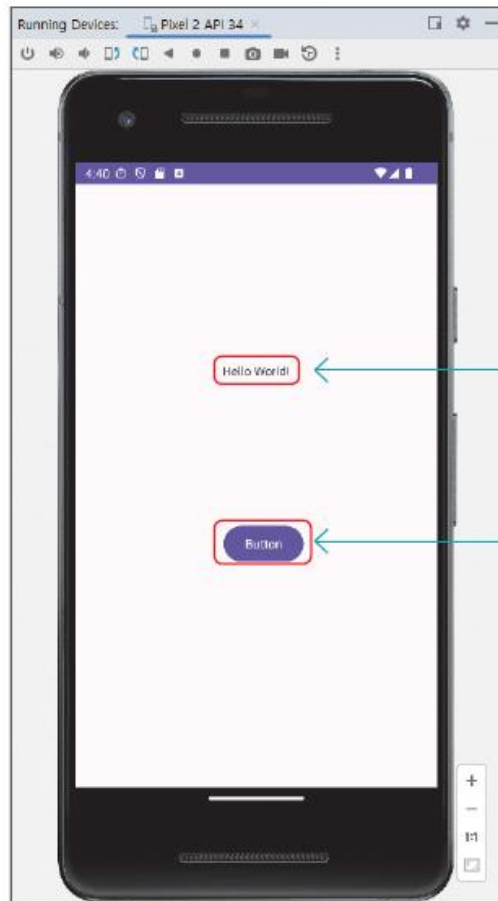




activity_main.xml 파일을 살펴보자.

- 비주얼 도구를 이용하여 화면을 제작하더라도 최종적으로는 XML 파일로 저장된다.





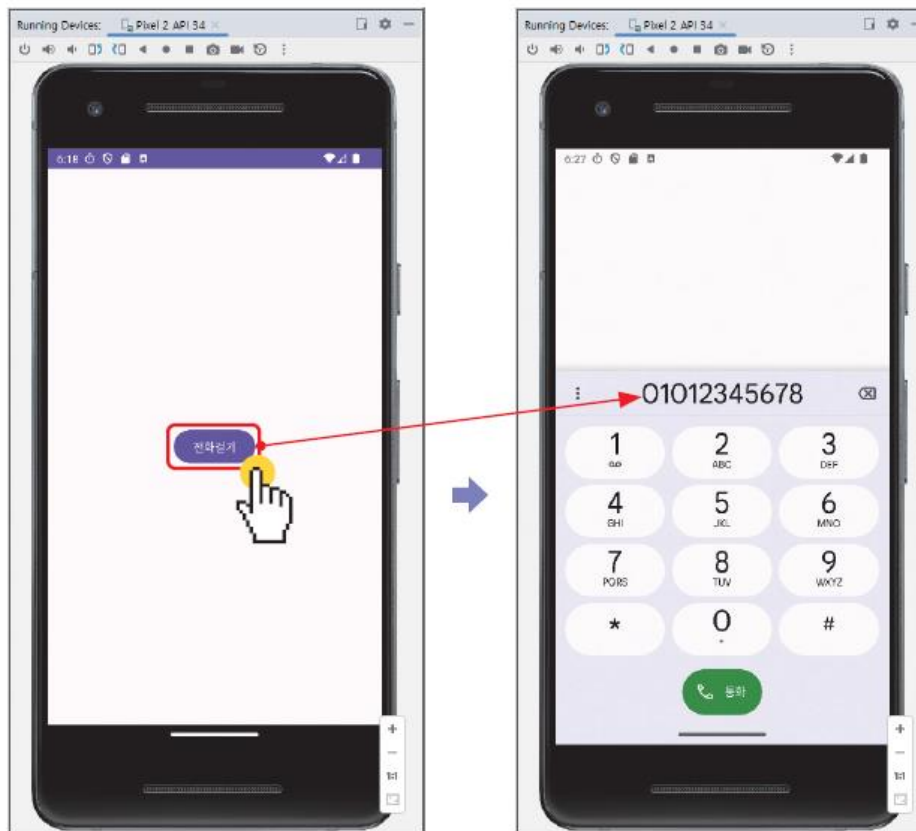
첫 번째 위젯인 TextView이다.

버튼을 눌러보자. 눌러지지만 아직까지 아무런 반응은 없는데 이것은 당연하다. 반응을 주려면 버튼의 이벤트를 자바 언어로 처리하여야 한다.



예제: 비주얼 도구 사용해보기 II

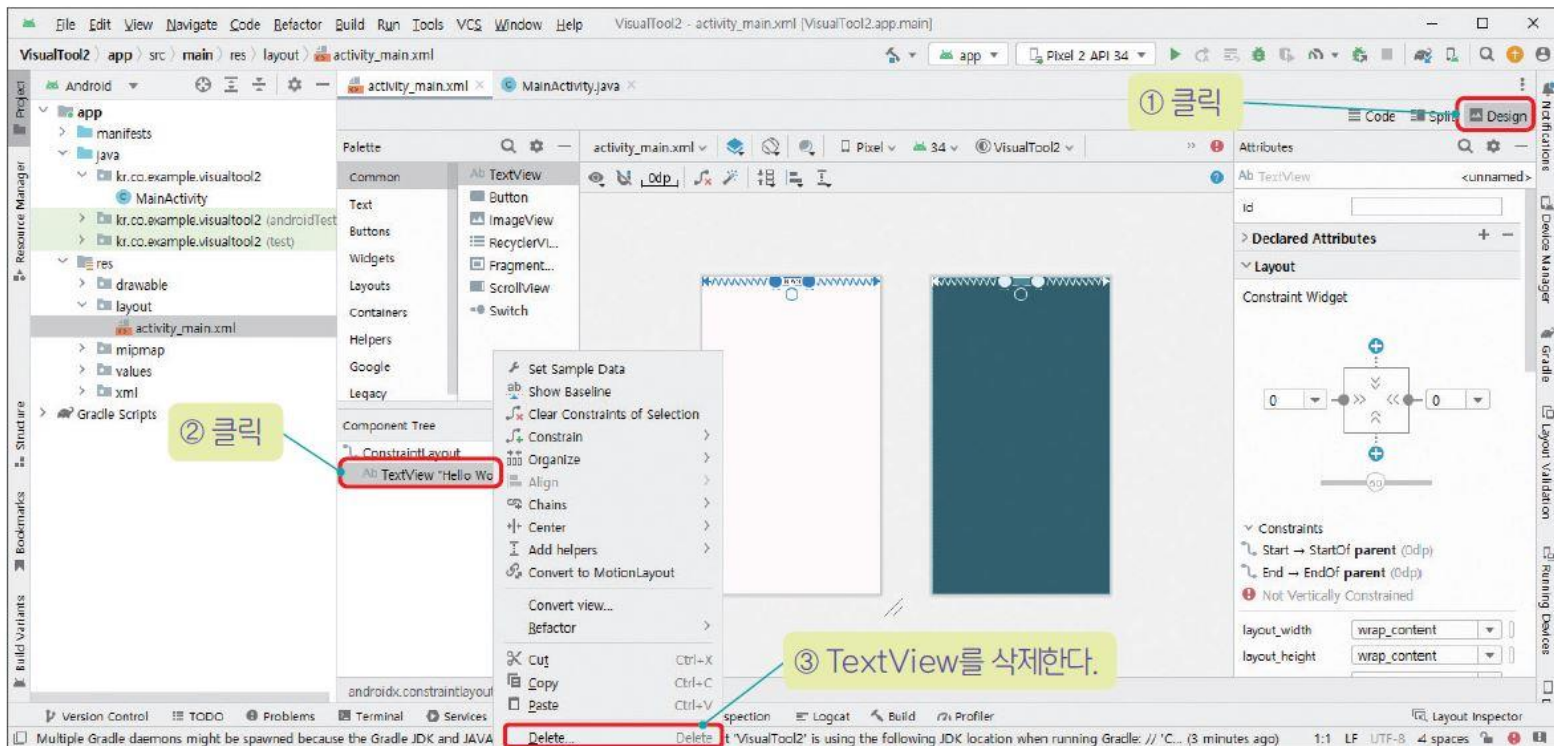
- 버튼을 누르면 전화 걸기 화면이 나오도록 하자.





Lab: 비주얼 도구 사용해보기 II

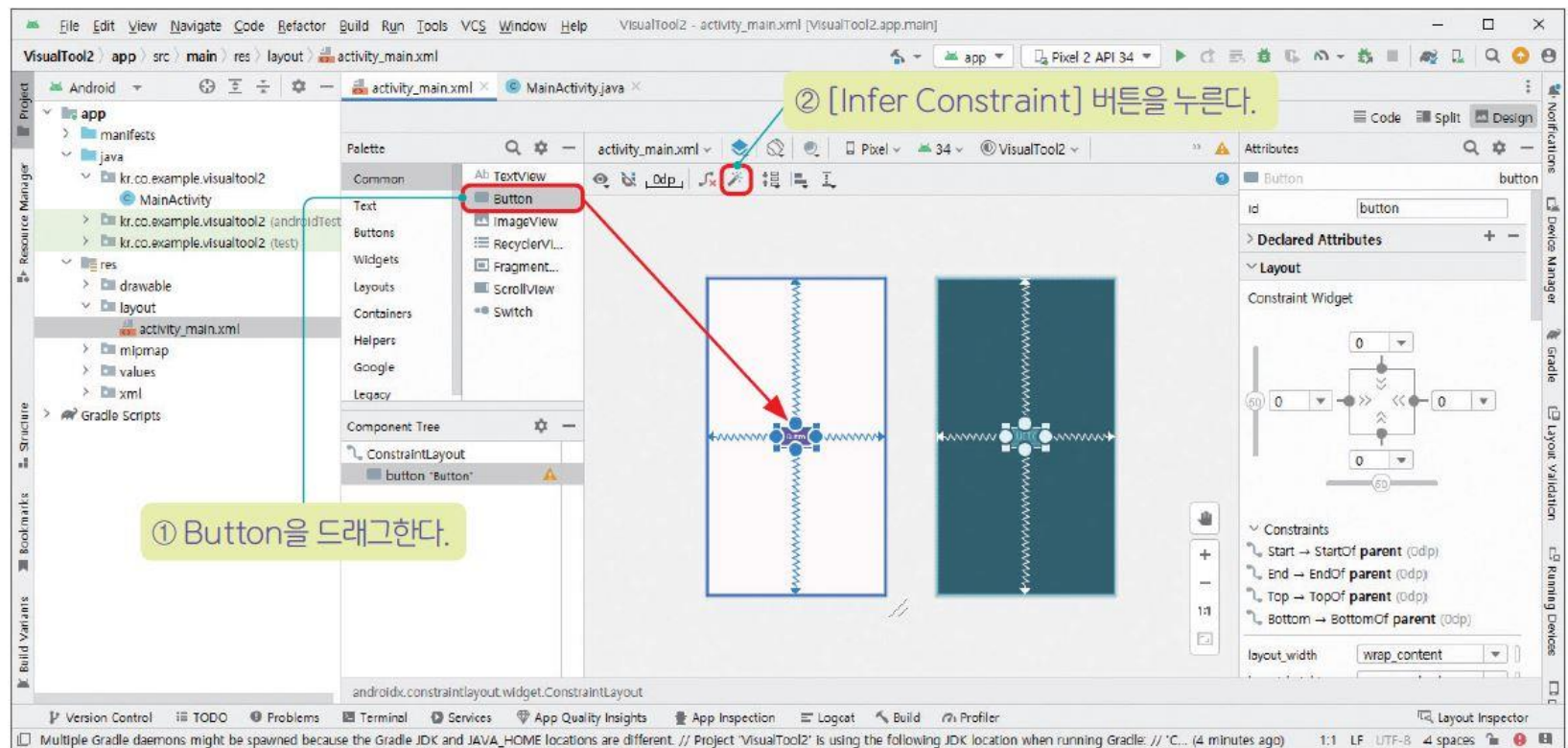
- 마우스로 TextView를 선택한 후에 삭제해보자.





Lab: 비주얼 도구 사용해보기 II

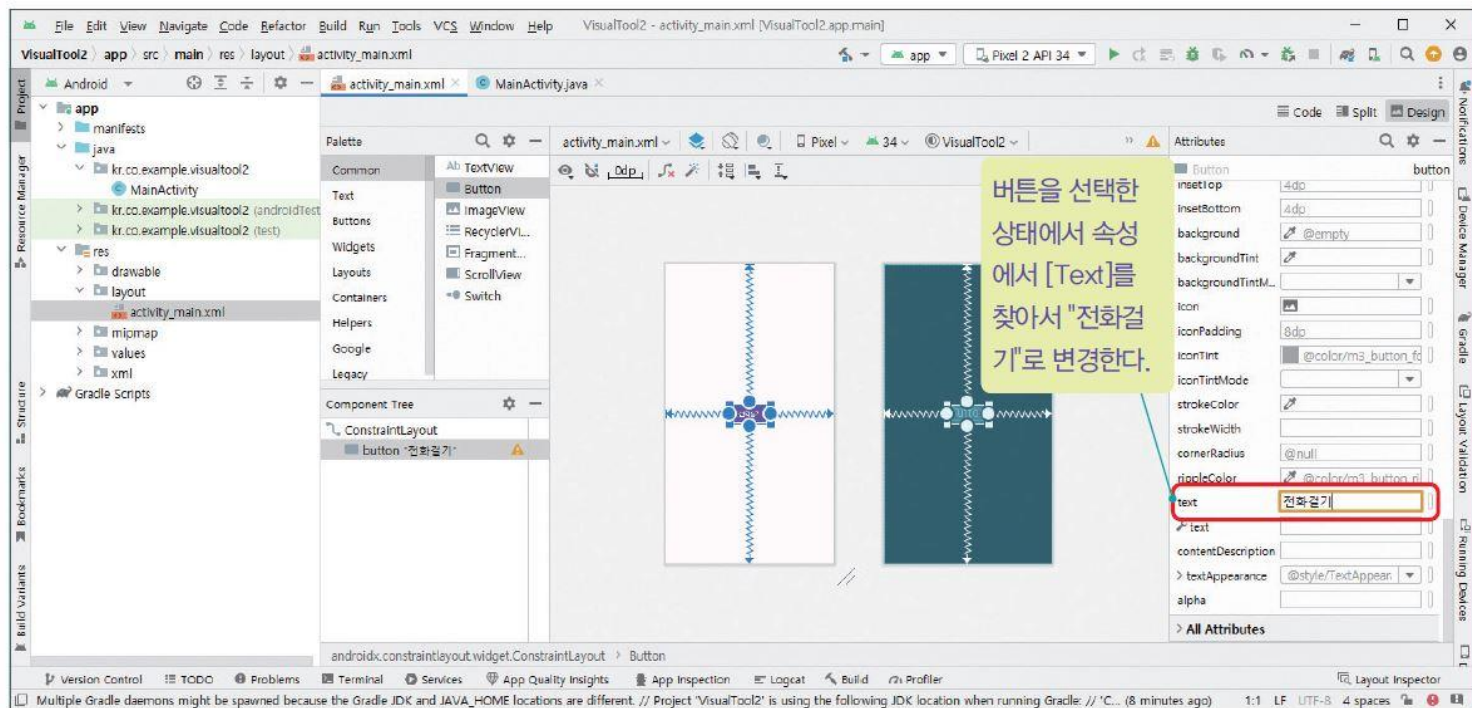
- 버튼을 끌고 온다.





Lab: 비주얼 도구 사용해보기 II

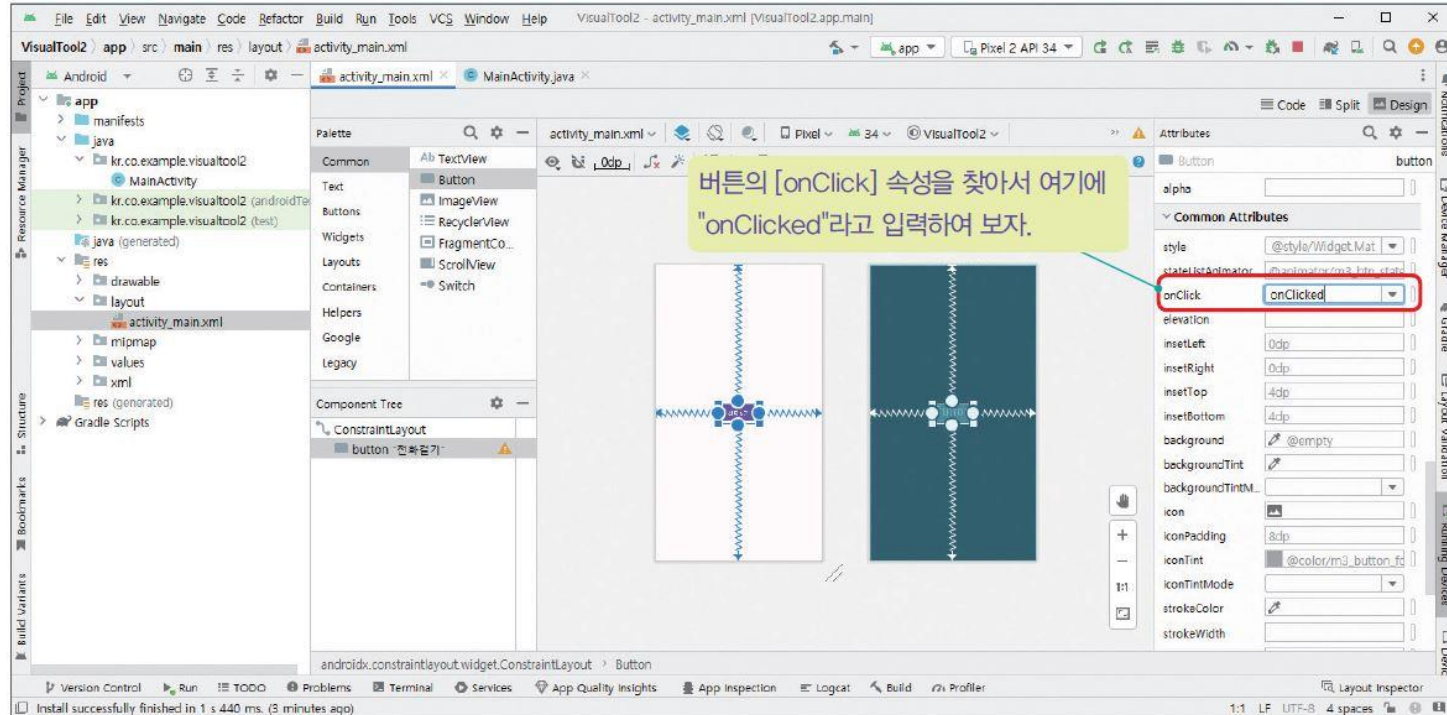
- 스크롤바를 움직여서 “text” 속성을 찾아서 “Button” 대신에 “전화걸기”로 변경하여 본다.





Lab: 비주얼 도구 사용해보기 II

- 버튼의 속성을 보여주는 창에서 “onClick” 속성을 찾아서 여기에 “onClicked”라고 입력하여 보자.





Lab: 비주얼 도구 사용해보기 II

- 우리는 `onClicked()`라고 하는 메소드를 자바 소스 파일에 정의하여 주어야 한다. 이번에는 `MainActivity.java` 파일을 열어서 다음 메소드를 추가해보자.

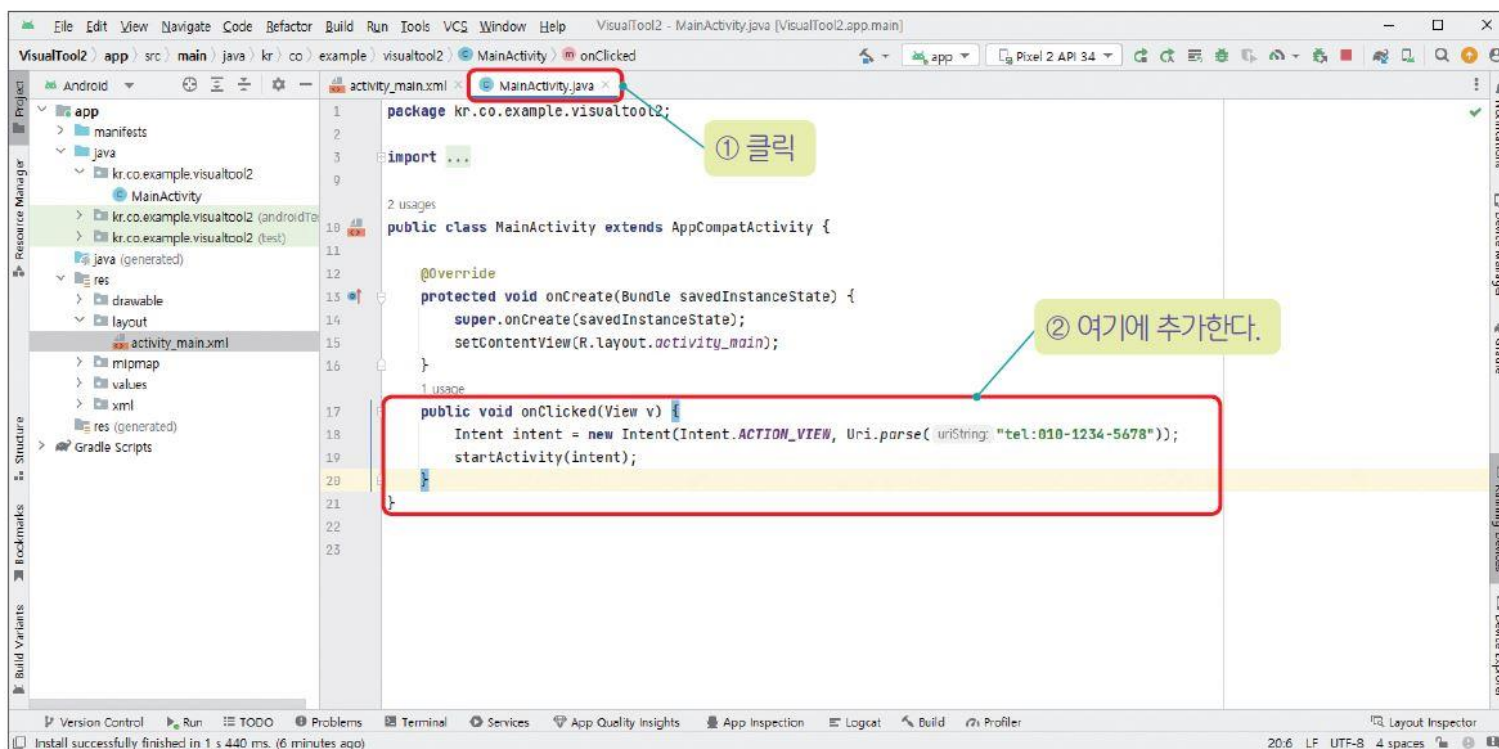
MainActivity.java

```
public void onClicked(View v) {  
    Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("tel:010-1234-5678"));  
    startActivity(intent);  
}
```



Lab: 비주얼 도구 사용해보기 II

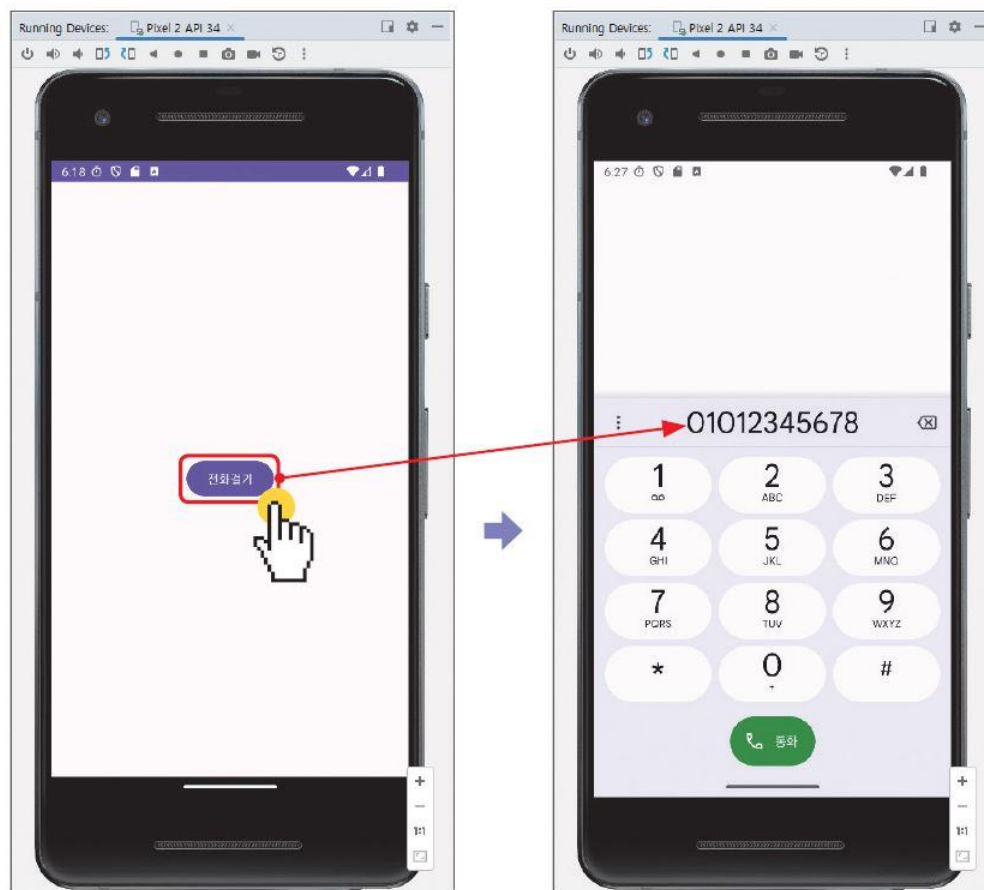
- 우리는 `onClicked()` 라고 하는 메소드를 자바 소스 파일에 정의하여 주어야 한다. 이번에는 `MainActivity.java` 파일을 열어서 다음 메소드를 추가해보자.





Lab: 비주얼 도구 사용해보기 II

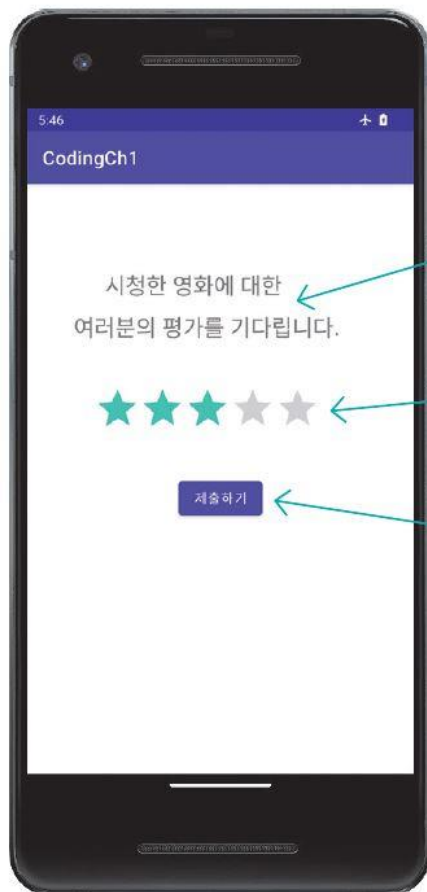
- 앱 실행





Coding Challenge:

비주얼 도구로 화면 만들어보기



[Text] → [Text View]를 마우스로 끌어서 만든다. 속성 중에서 [textSize]를 24sp로 설정한다.

[Widgets] → [Rating Bar]를 마우스로 끌어서 만든다.

[Buttons] → [Button]을 마우스로 끌어서 만든다.



Summary

- 애플리케이션은 컴포넌트들의 조합으로 만들어진다.
- 코드와 리소스는 철저하게 분리된다.
- 코드와 리소스는 개발 도구에 의하여 자동으로 생성되는 **R.java**에 의하여 서로 연결된다.



Q & A

