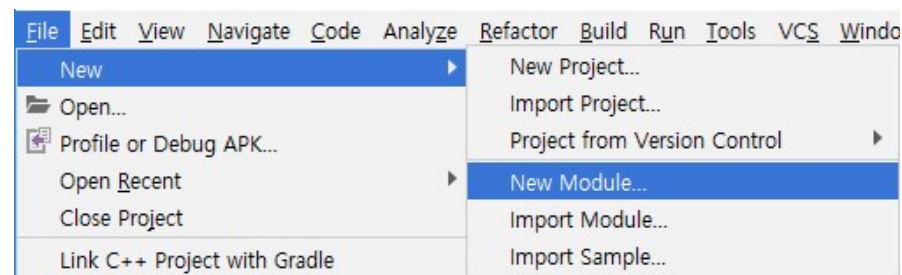
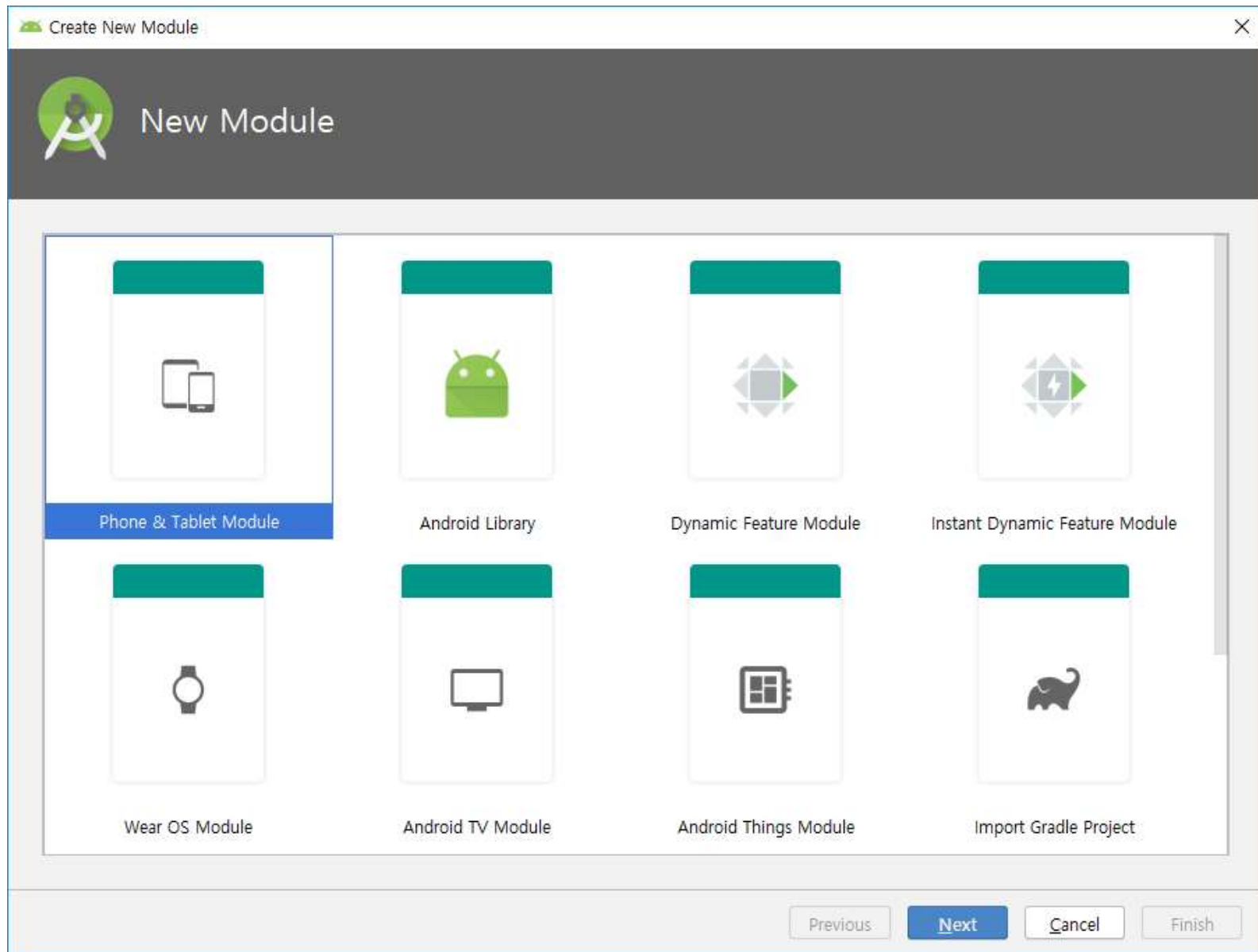




파일 다루기

Step 1 _ 모듈 생성





Create New Module

 Phone & Tablet Module

Configure the new module

Application/Library name

Module name

Package name

com.example.user.lab07

Edit

Minimum SDK

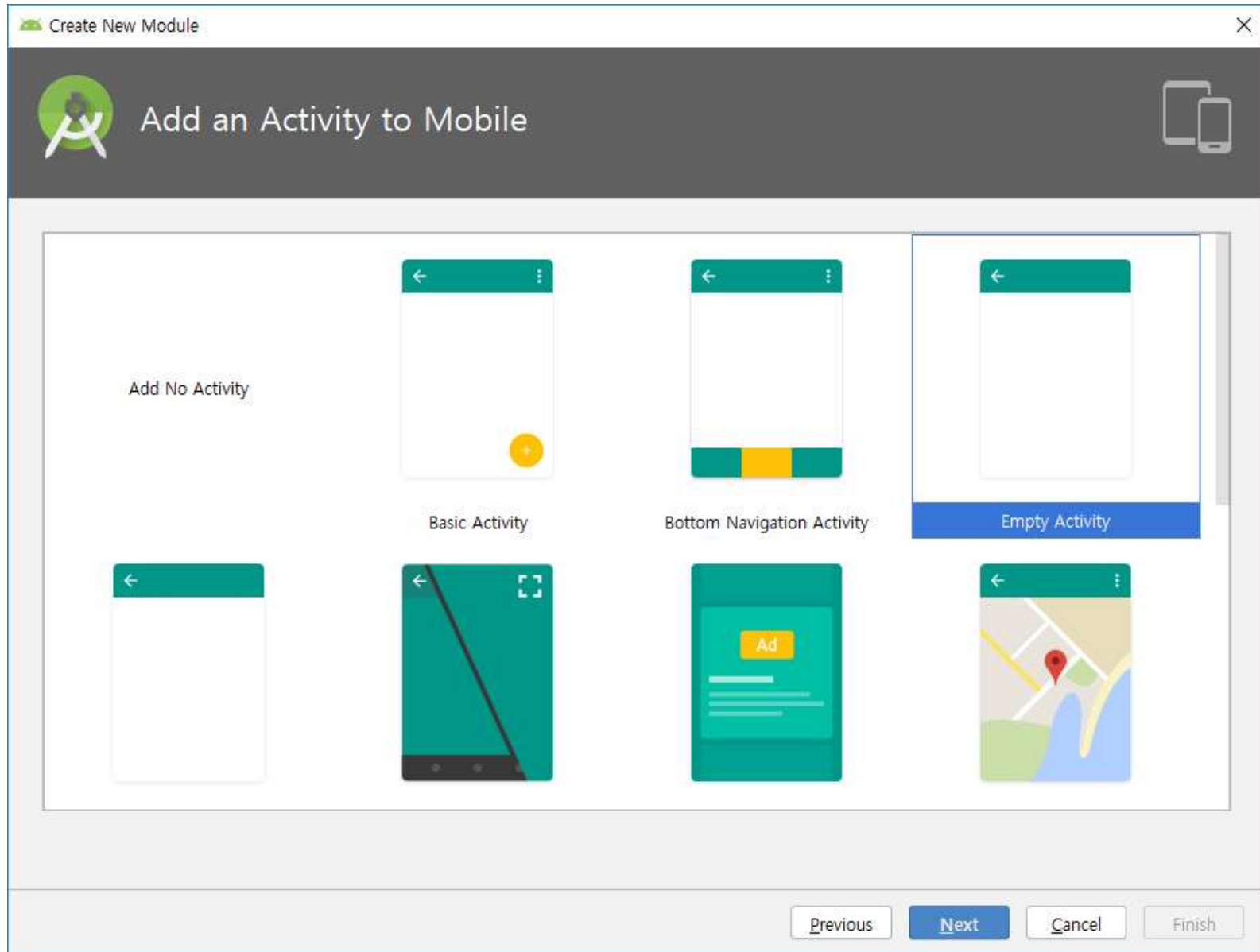
API 15: Android 4.0.3 (IceCreamSandwich) ▼

Previous



Next

Cancel

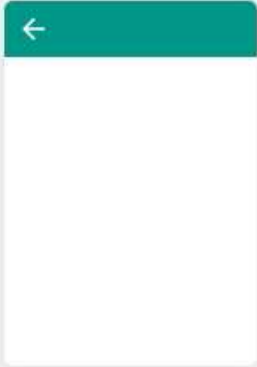
Finish



Create New Module

 Configure Activity

Creates a new empty activity



Activity Name:

MainActivity

☒ Generate Layout File

Layout Name:

activity_main

Source Language:

Java

The name of the activity class to create

Previous

Next

Cancel

Finish

Step 2 _ 퍼미션 부여



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.user.lab07">
```

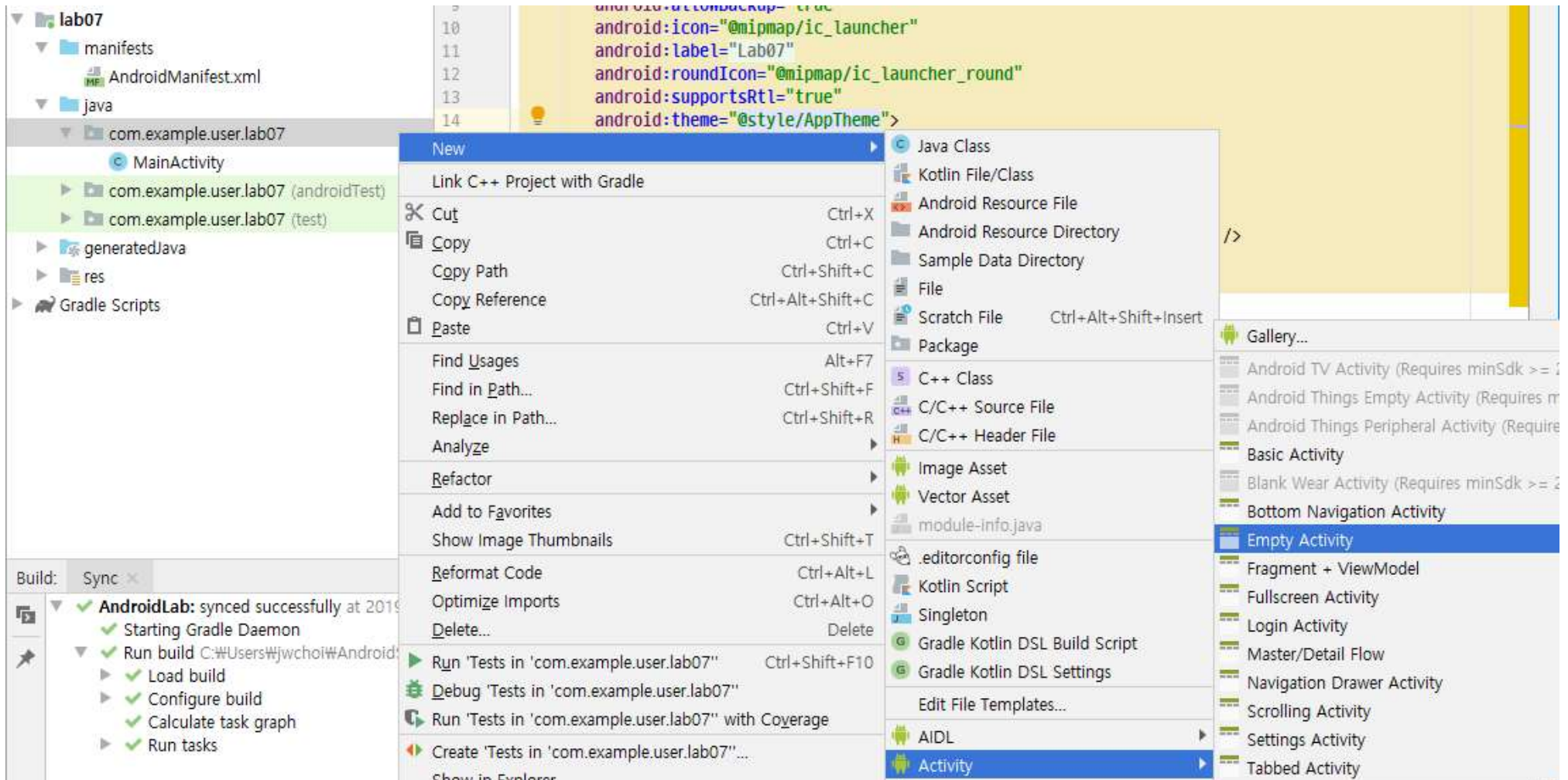
```
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="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>
```

Step 3 _ 결과 확인을 위한 액티비티 생성

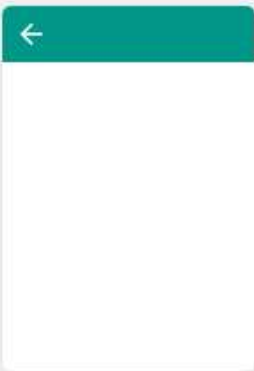


New Android Activity

 **Configure Activity**
Android Studio



Creates a new empty activity



Activity Name:

☒ Generate Layout File

Layout Name:

☐ Launcher Activity

Package name:

Source Language:

The name of the activity class to create

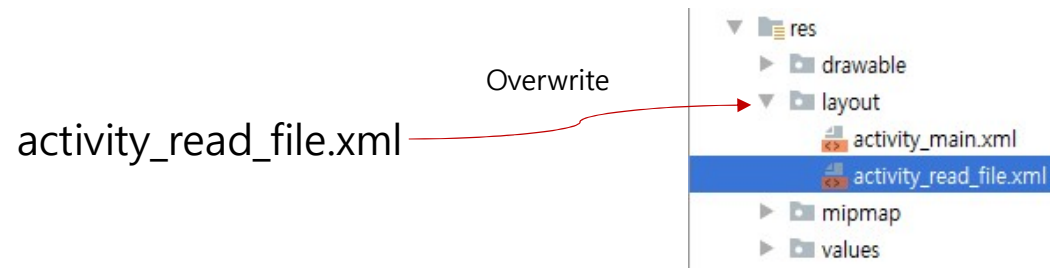
Previous

Next

Cancel

Finish

Step 4 _ activity_read_file.xml 복사



Step 5 _ ReadFileActivity 작성

```
public class ReadFileActivity extends AppCompatActivity {

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

        TextView textView = findViewById(R.id.fileResult);

        File file = new File(getFilesDir(), "myfile.txt");
        try {
            BufferedReader reader = new BufferedReader(new FileReader(file));
            StringBuffer buffer = new StringBuffer();
            String line;
            while((line = reader.readLine()) != null) {
                buffer.append(line);
            }
            textView.setText(buffer.toString());
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Step 5 _ ReadFileActivity 작성

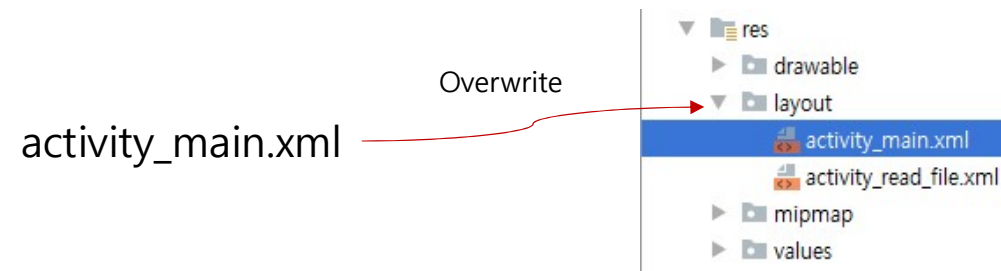
```
public class ReadFileActivity extends AppCompatActivity {

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

        TextView textView = findViewById(R.id.fileResult);

        File file = new File(Environment.getExternalStorageDirectory().getAbsolutePath()+"/myApp/myfile.txt");
        try {
            BufferedReader reader = new BufferedReader(new FileReader(file));
            StringBuffer buffer = new StringBuffer();
            String line;
            while((line = reader.readLine()) != null) {
                buffer.append(line);
            }
            textView.setText(buffer.toString());
            reader.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Step 6 _ activity_main.xml 복사



Step 7 _ MainActivity 작성

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    EditText contentView;
    Button btn;

    // 퍼미션 부여 여부
    boolean fileReadPermission;
    boolean fileWritePermission;

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

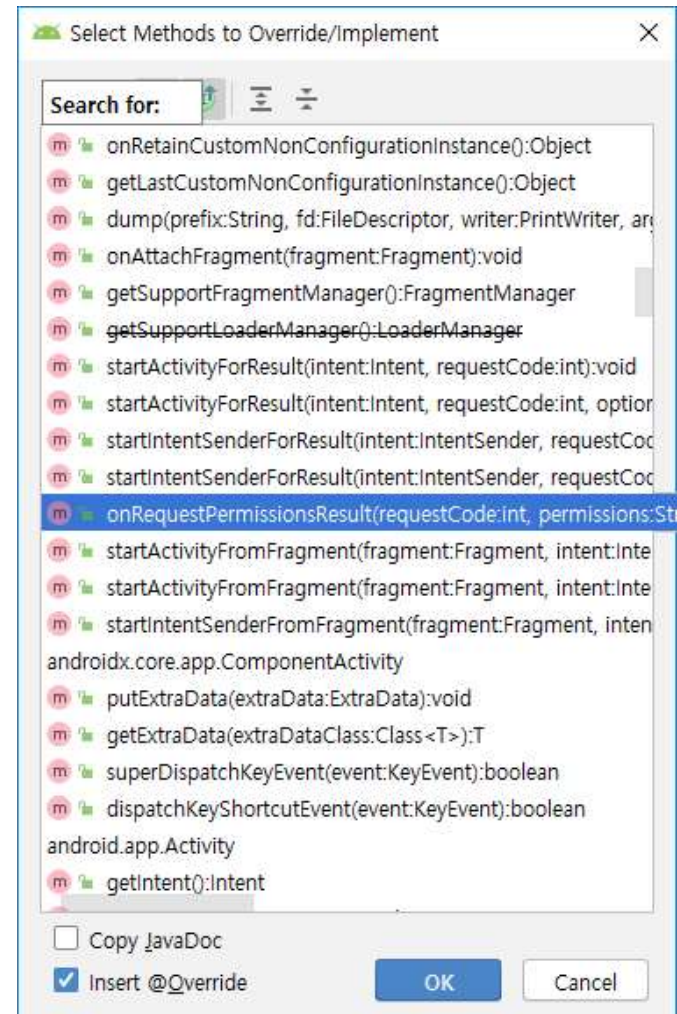
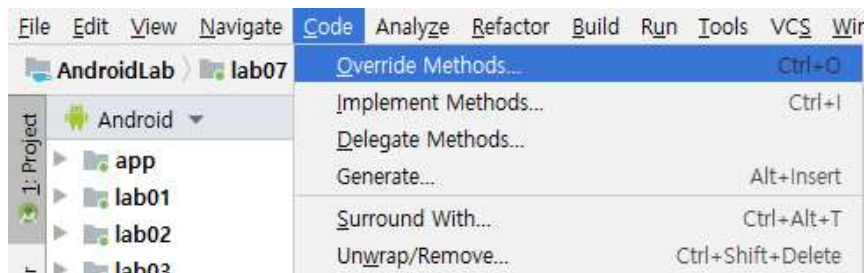
        contentView = findViewById(R.id.content);
        btn = findViewById(R.id.btn);

        btn.setOnClickListener(this);

        // permission 체크
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
            fileReadPermission=true;
        }
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
            fileWritePermission=true;
        }

        // permission 부여 안 될 경우 permission 요청
        if (!fileReadPermission || !fileWritePermission) {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.WRITE_EXTERNAL_STORAGE}, 200);
        }
    }
}
```

int: Application specific request code



If a local variable that contains a null value is passed as a parameter to a method with the @NonNull annotation attached to that parameter, building the code generates a **warning** indicating a non-null conflict.

```
// permission 부여 요청 결과 확인
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == 200 && grantResults.length > 0) {
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED)
            fileReadPermission = true;
        if (grantResults[1] == PackageManager.PERMISSION_GRANTED)
            fileWritePermission = true;
    }
}
```


내부 저장 공간 이용

```
@Override
public void onClick(View v) {
    String content = contentView.getText().toString();
    // 퍼미션이 부여되어 있다면
    if (fileReadPermission && fileWritePermission) {
        String filename = "myfile.txt";
        FileWriter writer;
        try {
            File file = new File(getFilesDir(), filename);
            // 파일이 없다면 새로 만들어 준다.
            if (!file.exists()) {
                file.createNewFile();
            }
            // file write
            writer = new FileWriter(file, true);
            writer.write(content);
            writer.flush();
            writer.close();

            // 결과 확인을 위한 FileReadActivity 실행 클래스
            Intent intent = new Intent(this, ReadFileActivity.class);
            // FileReadActivity로 화면 전환
            startActivity(intent);
        } catch (Exception e) {
            e.printStackTrace();
        }
    } else {
        showToast("permission 이 부여 안되어 기능 실행이 안됩니다.");
    }
}
```

android.content.Context의 메소드:
내부 파일이 저장된 파일 시스템 디렉터리의 절대 경로를 가져옵니다.

append - boolean if true, then data will be written to the end of
the file rather than the beginning.

```
private void showToast(String message) {
    Toast toast = Toast.makeText(this, message, Toast.LENGTH_SHORT);
    toast.show();
}
```

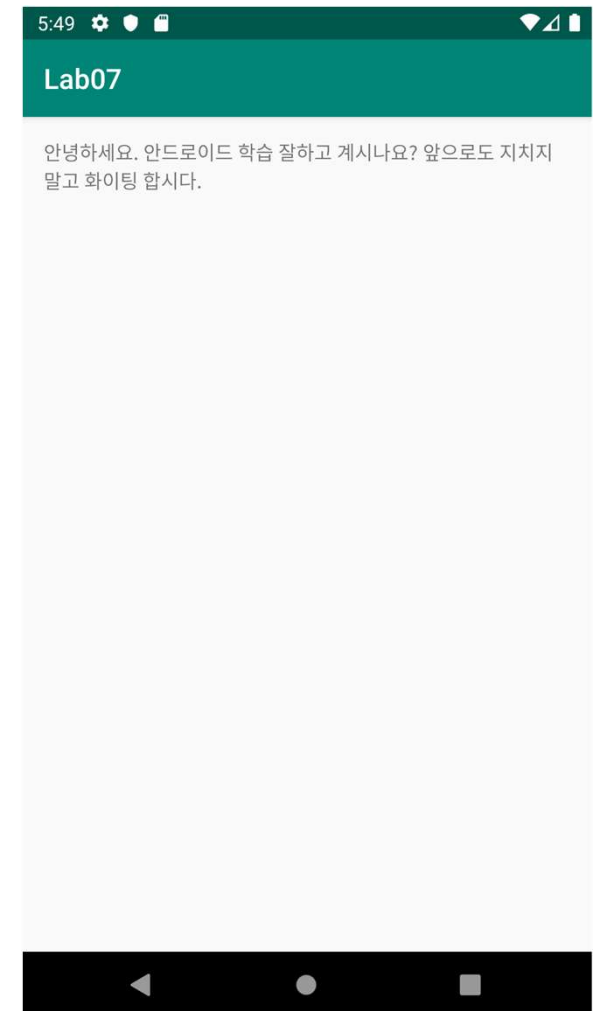
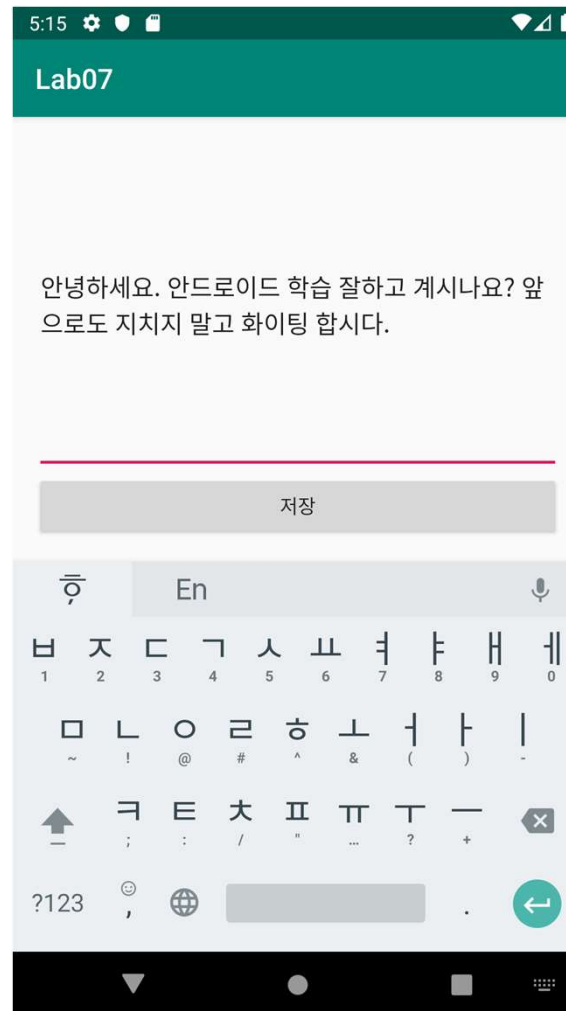
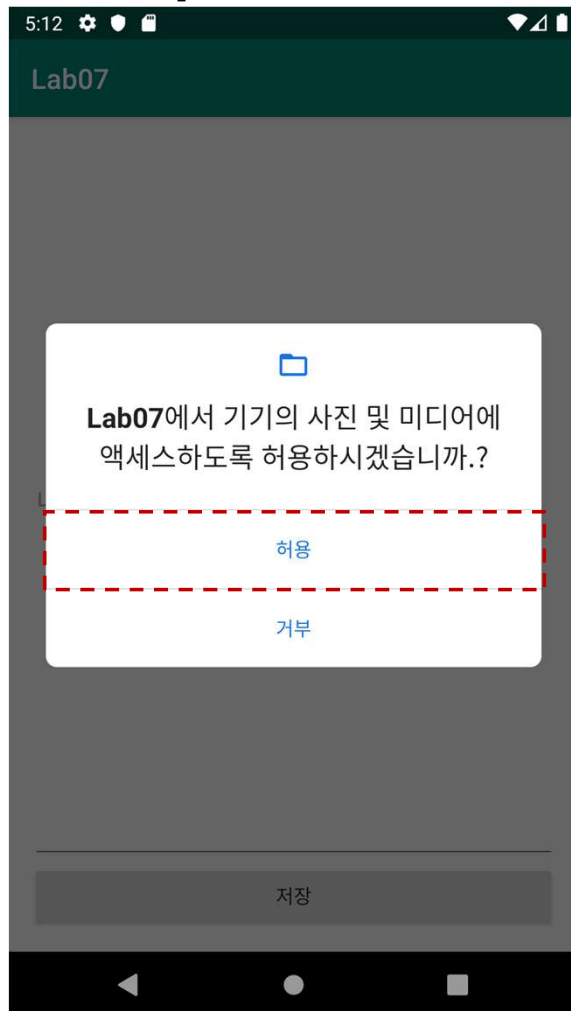
외부 저장 공간 이용

```
@Override
public void onClick(View v) {
    String content = contentView.getText().toString();
    // 퍼미션이 부여되어 있다면
    if (fileReadPermission && fileWritePermission) {
        FileWriter writer;
        try {
            // 외부 저장 공간 root 하위에 myApp이라는 폴더 경로 획득
            String dirPath = Environment.getExternalStorageDirectory().getAbsolutePath()+"/myApp";
            File dir = new File(dirPath);
            // 폴더가 없다면 새로 만들어 준다.
            if (!dir.exists()) {
                dir.mkdir();
            }
            // myApp 폴더 밑에 myfile.txt 파일 지정
            File file = new File(dir+"/myfile.txt");
            // 파일이 없다면 새로 만들어 준다.
            if (!file.exists()) {
                file.createNewFile();
            }
            // file write
            writer = new FileWriter(file, true);
            writer.write(content);
            writer.flush();
            writer.close();

            // 결과 확인을 위한 FileReadActivity 실행 클래스
            Intent intent = new Intent(this, ReadFileActivity.class);
            // FileReadActivity로 화면 전환
            startActivity(intent);
        } catch (Exception e) {
            e.printStackTrace();
        }
    } else {
        showToast("permission 이 부여 안되어 기능 실행이 안됩니다.");
    }
}
```

```
private void showToast(String message) {
    Toast toast = Toast.makeText(this, message, Toast.LENGTH_SHORT);
    toast.show();
}
```

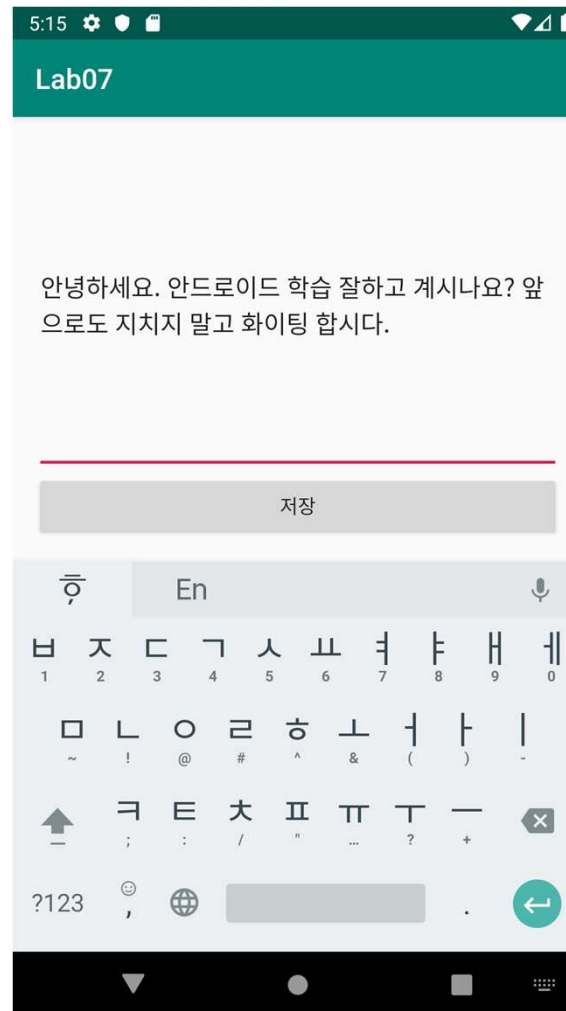
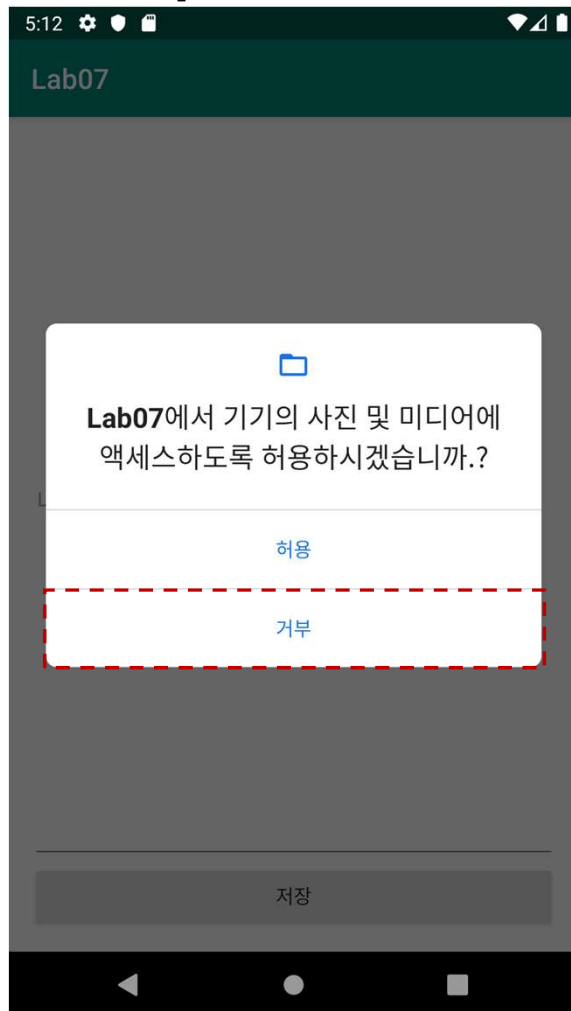
Step 8 _ 실행

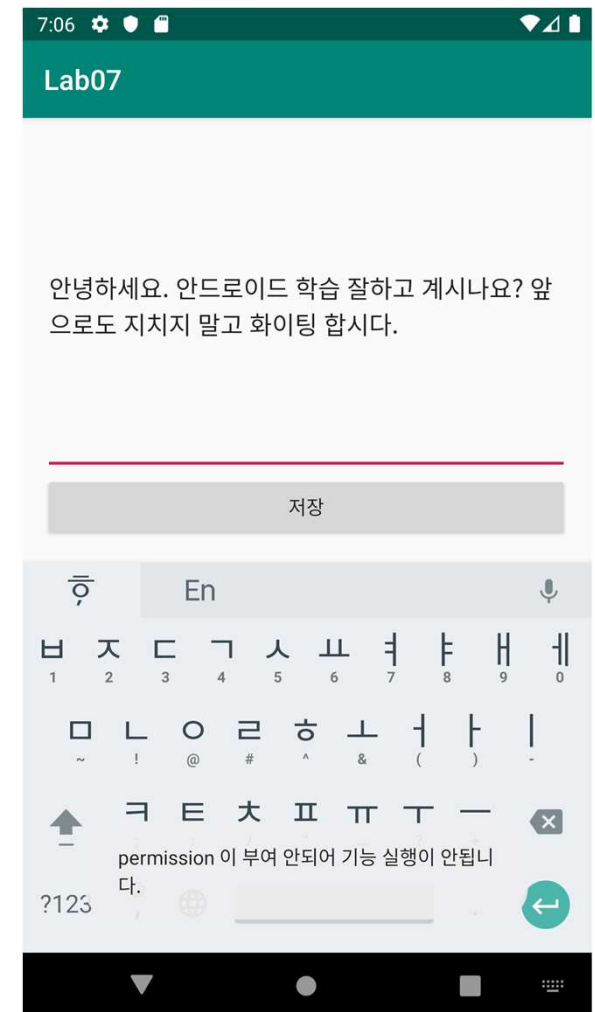
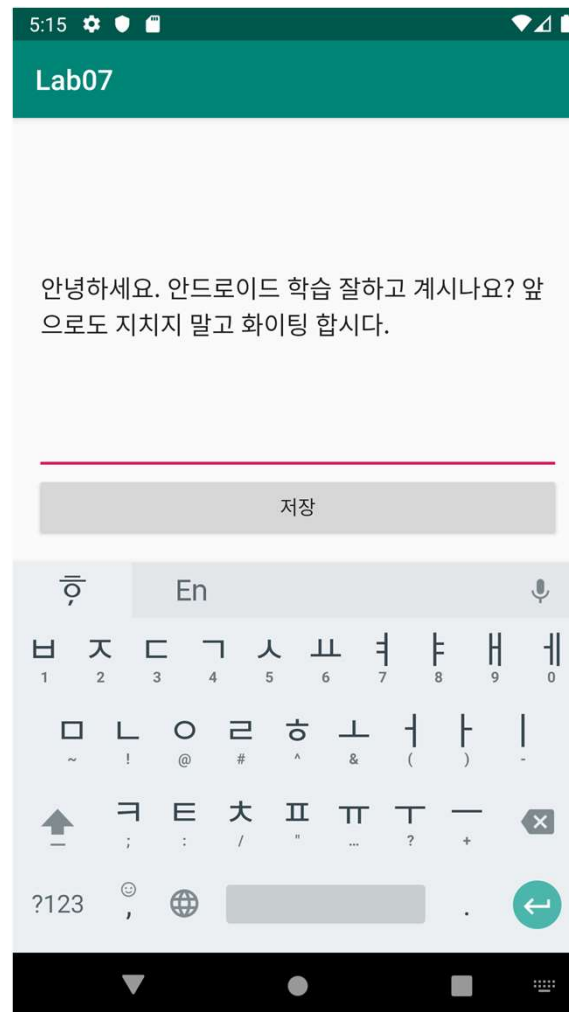
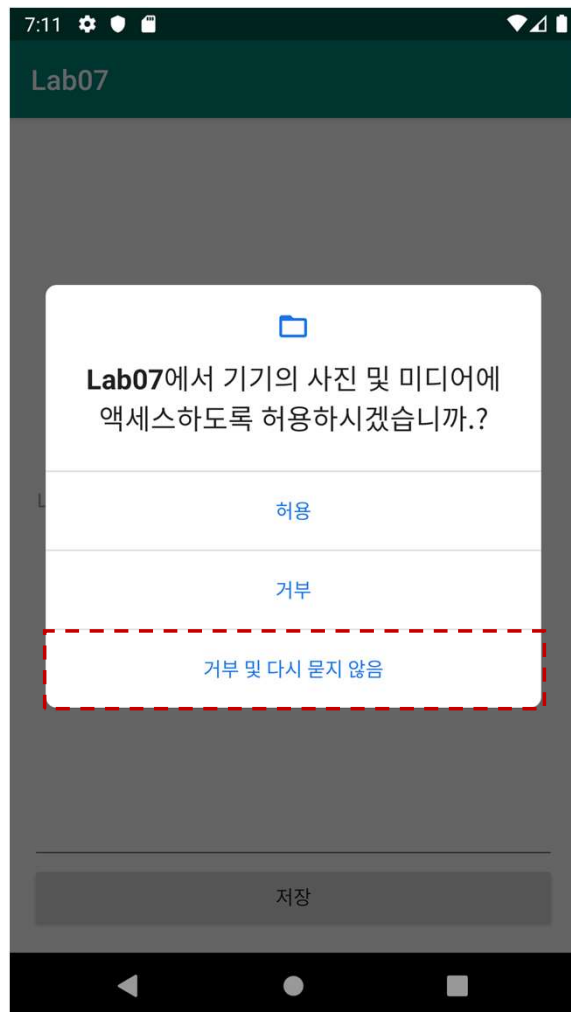


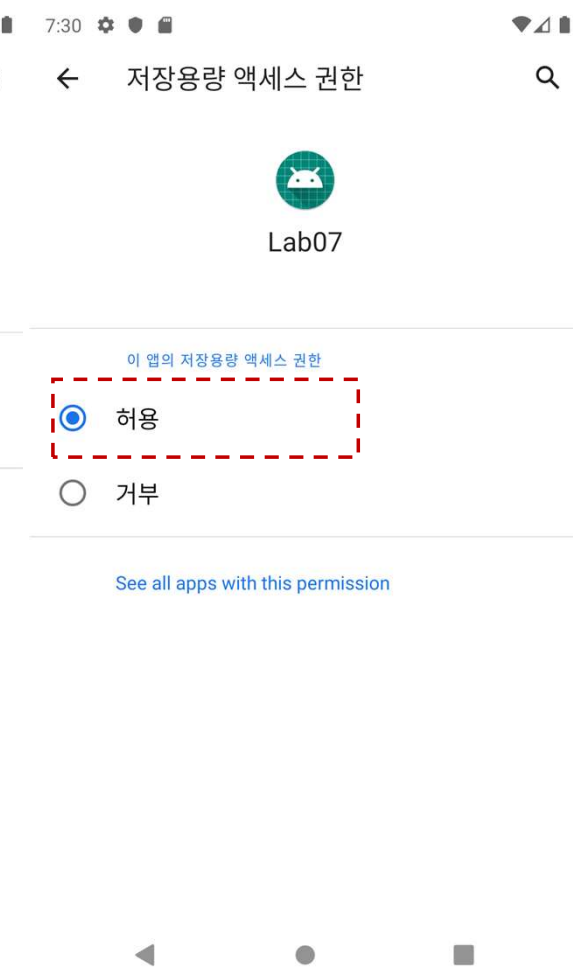
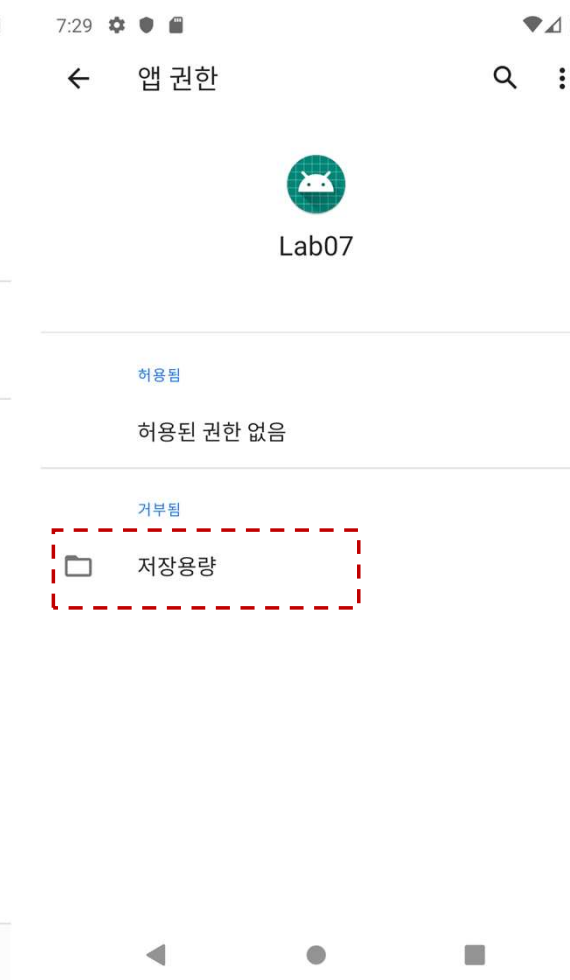
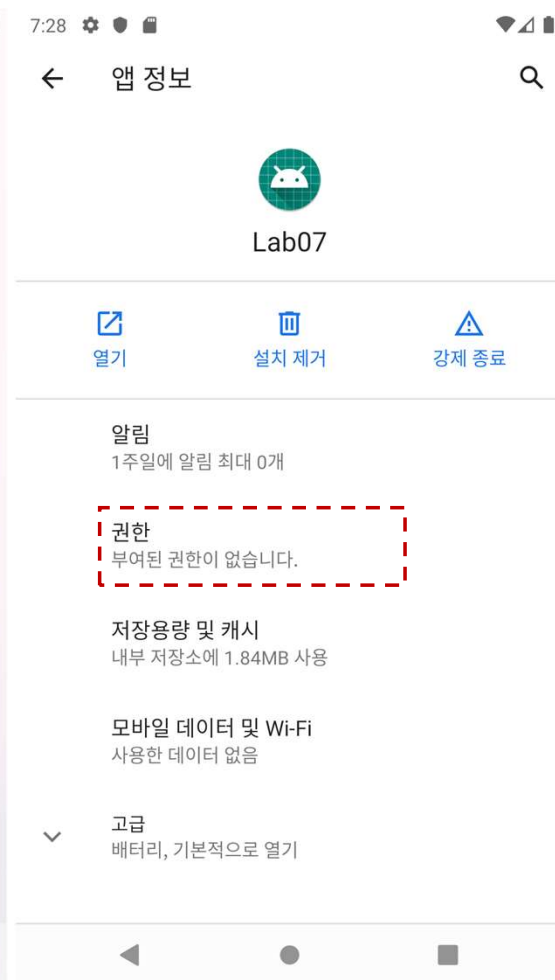
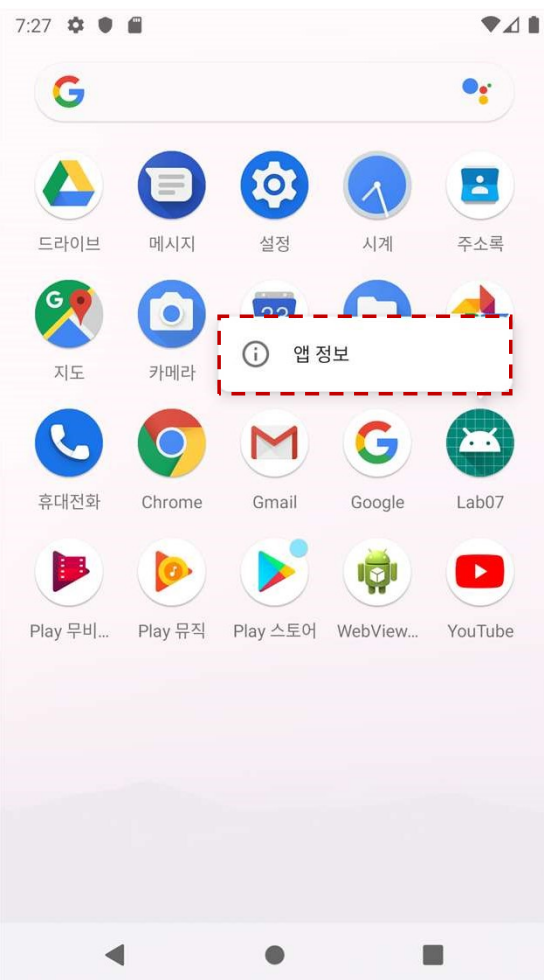
Device File Explorer			
Emulator Pixel_2_API_29 Android 10, API 29			
Name	Permi...	Date	Size
▶ acct	dr-xr-xr-	2019-07-23	0 B
▶ apex	drwxr-xr-	2019-07-23	320 B
▶ bin	lrw-r--r--	2009-01-01	11 B
▶ cache	drwxrwx-	2009-01-01	4 KB
▶ config	drwxr-xr-	2019-07-23	0 B
▶ d	lrw-r--r--	2009-01-01	17 B
▼ data	drwxrwx-	2019-07-11	4 KB
▶ app	drwxrwx-	2019-07-11	4 KB
▼ data	drwxrwx-	2019-07-11	4 KB
▶ android	drwxrwx-	2019-07-11	4 KB
▶ com.android.apps.t	drwxrwx-	2019-07-11	4 KB
▶ com.android.backu	drwxrwx-	2019-07-11	4 KB
▶ com.android.bips	drwxrwx-	2019-07-11	4 KB
▶ com.android.bluet	drwxrwx-	2019-07-11	4 KB
▶ com.android.bluet	drwxrwx-	2019-07-11	4 KB
▶ com.android.bookr	drwxrwx-	2019-07-11	4 KB
▶ com.android.callo	drwxrwx-	2019-07-11	4 KB
▶ com.android.camer	drwxrwx-	2019-07-11	4 KB
▶ com.android.captiv	drwxrwx-	2019-07-11	4 KB
▶ com.android.carrier	drwxrwx-	2019-07-11	4 KB
▶ com.android.carrier	drwxrwx-	2019-07-11	4 KB
▶ com.android.cellbr	drwxrwx-	2019-07-11	4 KB
▶ com.android.certin	drwxrwx-	2019-07-11	4 KB
▶ com.android.chror	drwxrwx-	2019-07-11	4 KB
▶ com.android.comp	drwxrwx-	2019-07-11	4 KB

Device File Explorer			
Emulator Pixel_2_API_29 Android 10, API 29			
Name	Per...	Date	...
▶ com.android.vending	drwxr	2019-07-	4 K
▶ com.android.vpndialogs	drwxr	2019-07-	4 K
▶ com.android.wallpaper.livepicker	drwxr	2019-07-	4 K
▶ com.android.wallpaperbackup	drwxr	2019-07-	4 K
▶ com.breel.wallpapers18	drwxr	2019-07-	4 K
▶ com.example.user.lab06	drwxr	2019-07-	4 K
▼ com.example.user.lab07	drwxr	2019-07-	4 K
▶ cache	drwxr	2019-07-	4 K
▶ code_cache	drwxr	2019-07-	4 K
▼ files	drwxr	2019-07-	4 K
myfile.txt	-rw---	2019-07-	139
▶ com.google.android.angle	drwxr	2019-07-	4 K
▶ com.google.android.apps.docs	drwxr	2019-07-	4 K
▶ com.google.android.apps.enterpr	drwxr	2019-07-	4 K
▶ com.google.android.apps.inputm	drwxr	2019-07-	4 K
▶ com.google.android.apps.maps	drwxr	2019-07-	4 K
▶ com.google.android.apps.messag	drwxr	2019-07-	4 K
▶ com.google.android.apps.nexusla	drwxr	2019-07-	4 K
▶ com.google.android.apps.photos	drwxr	2019-07-	4 K
▶ com.google.android.apps.pixelmi	drwxr	2019-07-	4 K
▶ com.google.android.apps.restore	drwxr	2019-07-	4 K
▶ com.google.android.apps.wallpap	drwxr	2019-07-	4 K
▶ com.google.android.apps.wallpap	drwxr	2019-07-	4 K
▶ com.google.android.calendar	drwxr	2019-07-	4 K
▶ com.google.android.configupdat	drwxr	2019-07-	4 K

Step 9 _ 앱 삭제 후 실행







Permissions overview

- The purpose of a permission is to protect the privacy of an Android user.
- Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet).
- Protection levels: The protection level affects whether runtime permission requests are required.
 - Normal permissions
 - Dangerous permissions

<https://developer.android.com/guide/topics/permissions/overview>

Dangerous permissions and permission groups



사용자 권한 화면에는 퍼미션 그룹 단위로 알려줌

퍼미션 그룹	퍼미션
CALENDAR	READ_CALENDAR
	WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS
	WRITE_CONTACTS
	GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION
	ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE
	CALL_PHONE
	READ_CALL_LOG
	WRITE_CALL_LOG
	ADD_VOICEMAIL
	USE_SIP
	PROCESS_OUTGOING_CALLS
	BODY_SENSORS
SENSORS	BODY_SENSORS
SMS	SEND_SMS
	RECEIVE_SMS
	READ_SMS
	RECEIVE_WAP_PUSH
	RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE
	WRITE_EXTERNAL_STORAGE

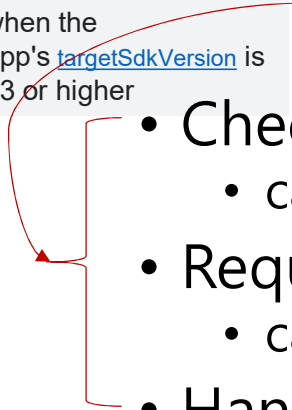
이용빈도가 높은 퍼미션

- ACCESS_FINE_LOCATION: 정확한 위치 정보 액세스
- ACCESS_NETWORK_STATE: 네트워크에 대한 정보 액세스
- ACCESS_WIFI_STATE: 와이파이 네트워크에 대한 정보 액세스
- BATTERY_STATS: 배터리 통계 수집
- BLUETOOTH: 연결된 블루투스 장치에 연결
- BLUETOOTH_ADMIN: 블루투스 장치를 검색하고 페어링
- CALL_PHONE: 다이얼 UI를 거치지 않고 전화를 시작
- CAMERA: 카메라 장치에 액세스
- INTERNET: 네트워크 연결
- READ_CONTACTS: 사용자의 연락처 데이터 읽기
- READ_EXTERNAL_STORAGE: 외부 저장소에서 파일 읽기
- READ_PHONE_STATE: 장치의 전화번호, 네트워크 정보, 진행 중인 통화 상태 등 전화 상태에 대한 읽기
- READ_SMS: SMS 메시지 읽기
- RECEIVE_BOOT_COMPLETED: 부팅 완료 시 수행
- RECEIVE_SMS: SMS 메시지 수신
- RECORD_AUDIO: 오디오 녹음
- SEND_SMS: SMS 메시지 발신
- VIBRATE: 진동 울리기
- WRITE_CONTACTS: 사용자의 연락처 데이터 쓰기
- WRITE_EXTERNAL_STORAGE: 외부 저장소에 파일 쓰기

Request App Permissions

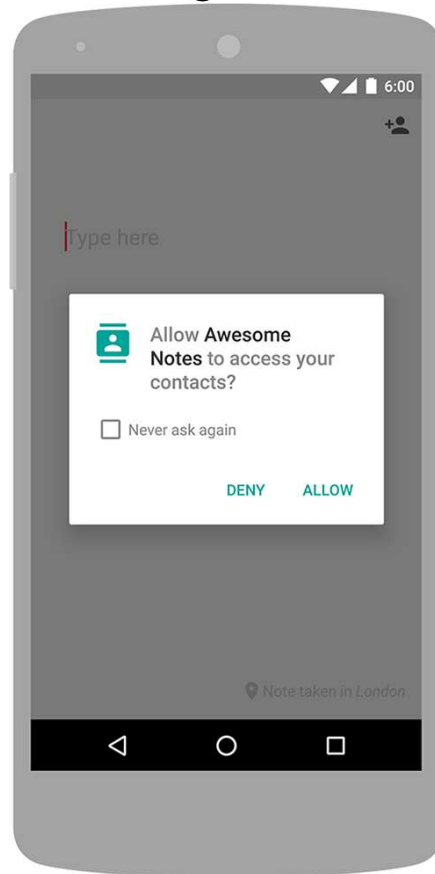
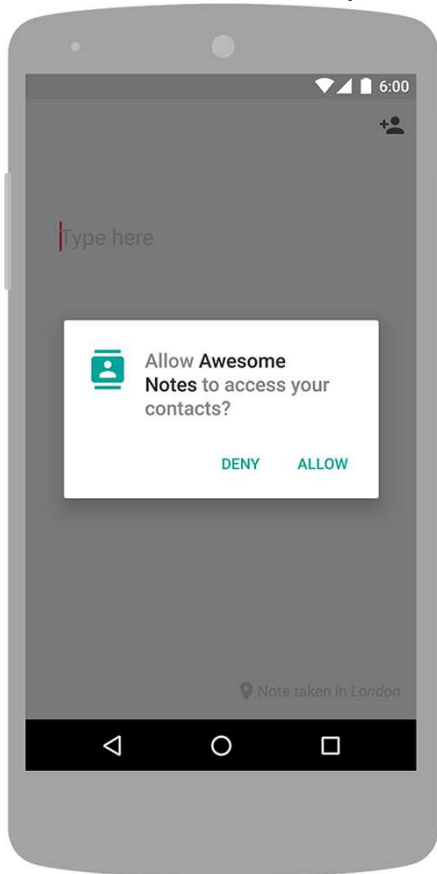
- Add permissions to the manifest
 - put a `<uses-permission>` element in your app manifest
 - Normal permissions
 - the system immediately grants them upon installation
 - Dangerous permissions
 - the user must explicitly grant your app access
- Check for permissions
 - call the `checkSelfPermission()` method
- Request permissions
 - call the `requestPermissions()` method
- Handle the permissions request response
 - override the callback(`onRequestPermissionsResult()`) method

when the
app's `targetSdkVersion` is
23 or higher

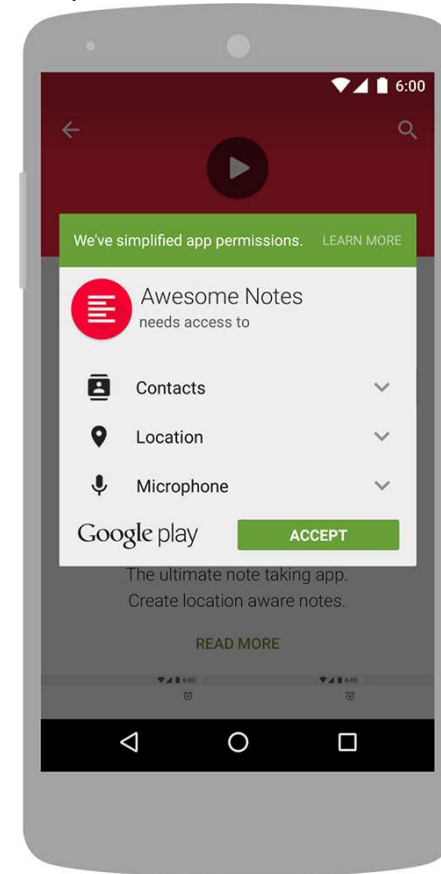


Request prompts for dangerous permissions

- Runtime requests (Android 6.0 and higher)



- Install-time requests (Android 5.1.1 and below)



파일에 읽고 쓰기

- 안드로이드에서 파일 관련 프로그램은 대부분 자바 API(java.io 패키지의 클래스들)를 그대로 사용
 - File: 파일 및 디렉토리를 지칭하는 클래스
 - FileInputStream: 파일에서 바이트 데이터를 읽기 위한 함수 제공
 - FileOutputStream: 파일에 바이트 데이터를 쓰기 위한 함수 제공
 - FileReader: 파일에서 문자열 데이터를 읽기 위한 함수 제공
 - FileWriter: 파일에 문자열 데이터를 쓰기 위한 함수 제공
- 안드로이드 스마트폰의 파일 저장 공간은 내부와 외부로 구분하여 생각해야 함
 - 외부 저장 공간
 - 모든 기기에서 외부 저장 공간을 제공한다고 보장할 수 없음
 - 기기에 따라 SD 카드와 같이 이동식 저장 장치로 제공하는 경우
 - 이동식 저장 장치 없이 영구 저장소에 내부, 외부 파티션을 나누어 항상 두 개의 저장 공간을 제공하는 경우
 - 이 공간은 모든 앱이 접근하여 파일을 이용할 수 있음
 - 내부 저장 공간
 - 어느 스마트폰에서나 제공하므로 항상 이용할 수 있음
 - 앱을 설치할 때 해당 앱을 위한 저장 공간이 할당
 - 이 공간은 앱이 삭제되면 함께 삭제
 - 이 공간에 저장한 파일은 해당 앱에서만 접근할 수 있음

Environment 클래스

- 안드로이드에서 저장소와 관련된 각종 정보는 Environment 클래스로 얻을 수 있음
 - 외부 저장 공간에 대한 정보 획득

```
String state = Environment.getExternalStorageState();
if (state.equals(Environment.MEDIA_MOUNTED)) {
    if (state.equals(Environment.MEDIA_MOUNTED_READ_ONLY)) {
        externalStorageReadable = true;
        externalStorageWritable = false;
    } else {
        externalStorageReadable = true;
        externalStorageWritable = true;
    }
} else {
    externalStorageReadable = externalStorageWritable = false;
}
```

<https://developer.android.com/reference/android/os/Environment>

Environment 클래스

- 안드로이드에서 저장소와 관련된 각종 정보는 Environment 클래스로 얻을 수 있음

- 외부 저장 공간 경로
 - 개발자가 지정한 폴더

```
String dirPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/myApp";
```

- 스마트 폰에 설정된 공용 폴더

```
File file1 = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), "a.jpg");
```

- Environment.DIRECTORY_ALARMS: 알람으로 사용할 오디오 파일 저장 폴더
- Environment.DIRECTORY_DCIM: 카메라로 촬영한 사진 저장 폴더
- Environment.DIRECTORY_DOWNLOADS: 다운로드한 파일 저장 폴더
- Environment.DIRECTORY_MUSIC: 음악 파일 저장 폴더
- Environment.DIRECTORY_MOVIES: 영상 파일 저장 폴더
- Environment.DIRECTORY_NOTIFICATIONS: 알림음으로 사용할 오디오 파일 저장 폴더
- Environment.DIRECTORY_PICTURES: 이미지 파일 저장 폴더

<https://developer.android.com/reference/android/os/Environment>

SharedPreferences

```
public interface SharedPreferences  
  
android.content.SharedPreferences
```

- 데이터를 간단하게 키-값(key-value) 쌍으로 저장
- 파일(XML)로 저장
 - 개발자가 직접 파일을 읽고 쓰는 코드를 작성하지 않음
 - SharedPreferences 객체를 이용

<https://developer.android.com/training/data-storage/shared-preferences>

Get a handle to shared preferences

- Activity의 `getPreferences(int mode)`

```
SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
```

- 액티비티 이름을 파일명으로 사용
 - 예) MainActivity -> MainActivity.xml
- 하나의 액티비티만을 위한 저장 공간
 - 다른 액티비티에서는 데이터를 이용할 수 없음

- Context의 `getSharedPreferences(String name, int mode)`

```
SharedPreferences sharedPref = getSharedPreferences("my_prefs", Context.MODE_PRIVATE);
```

- 앱 내 다른 액티비티나 컴포넌트들이 데이터를 공유할 수 있음

- `PreferenceManager.getDefaultSharedPreferences(Context context)`

```
SharedPreferences sharedPref= PreferenceManager.getDefaultSharedPreferences(this);
```

- 앱의 패키지명을 파일명으로 사용
 - 예) com.example.test -> com.example.test_preferences
- 앱 내 다른 액티비티나 컴포넌트들이 데이터를 공유할 수 있음

MODE_PRIVATE: 자기 앱 내에서 사용. 외부 앱에서 접근 불가
MODE_WORLD_READABLE: 외부 앱에서 읽기 가능
MODE_WORLD_WRITEABLE: 외부 앱에서 쓰기 가능

Write to shared preferences

- 데이터를 저장하려면 Editor 클래스의 함수를 이용

```
SharedPreferences.Editor editor=sharedPref.edit();  
editor.putString( "data1", "hello" );  
editor.putInt( "data2", 100);  
editor.commit();
```

- 저장한 데이터를 최종 반영하기 위해 commit() 함수 호출

```
putBoolean(String key, boolean value)  
putFloat(String key, float value)  
putInt(String key, int value)  
putLong(String key, long value)  
putString(String key, String value)
```

Read from shared preferences

- 데이터의 획득

```
String data1=sharedPref.getString("data1", "none");  
int data2=sharedPref.getInt("data2", 0);
```

```
getBoolean(String key, boolean defValue)  
getFloat(String key, float defValue)  
getInt(String key, int defValue)  
getLong(String key, long defValue)  
getString(String key, String defValue)
```