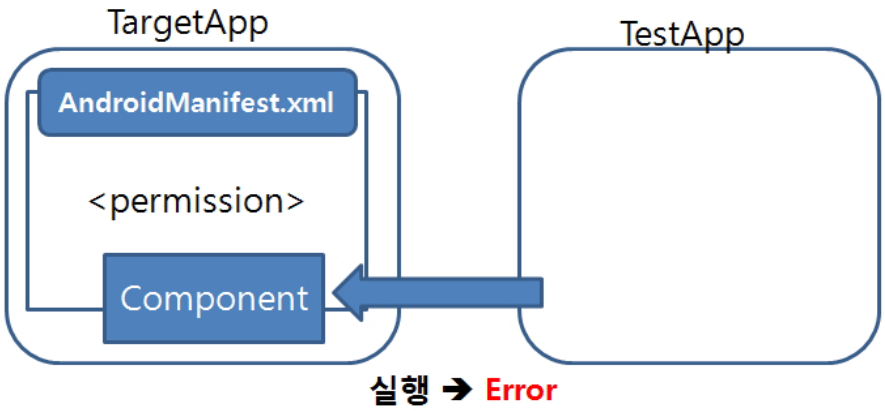
9장. 파일 및 SharedPreference을 이용한 데이터 영속화

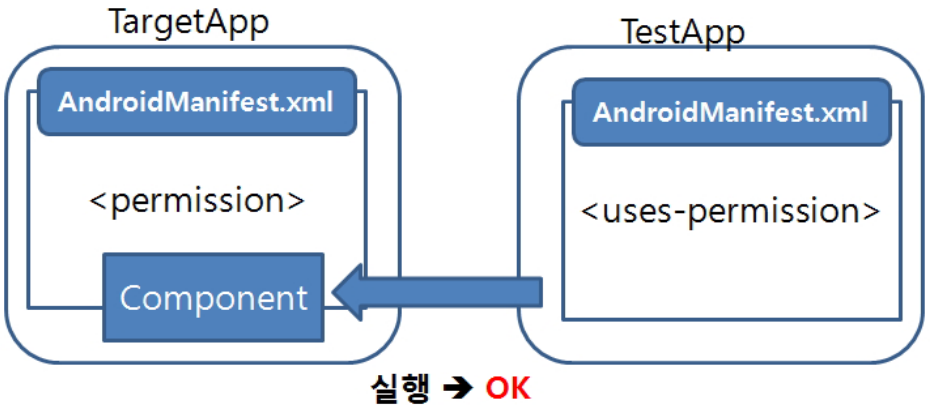
9.1 퍼미션

9.1.1. 퍼미션이란?

- AndroidManifest.xml에 들어가는 설정
- 어떤 앱이 <permission>을 부여했다면 그 앱을 이용하는 앱은 <uses-permission>을 선언

- <permission> 이용





9.1 퍼미션

- AndroidManifest.xml에 <permission> 태그를 추가

```
<permission android:name="com.test.permission.SOME_PERMISSION"
    android:label="SOME Permission"
    android:description="@string/permission"
    android:protectionLevel="normal"/>
```

- name: 퍼미션의 이름
- label, description: 퍼미션에 대한 설명(사용자에게 보이는 문자열)
- protectionLevel: 보호 수준

- protectionLevel을 이용해 보호 수준

- normal: 낮은 수준의 보호. 사용자에게 권한 부여 요청이 필요 없는 경우
- dangerous: 높은 수준의 보호. 사용자에게 권한 부여 요청이 필요한 경우
- signature: 동일한 키로 사인된 앱만 실행
- signatureOrSystem: 안드로이드 시스템 앱이거나 동일 키로 사인된 앱만 실행

9.1 퍼미션

- 컴포넌트에 퍼미션을 적용

```
<activity android:name=".SomeActivity"
    android:permission="com.test.permission.SOME_PERMISSION">
    <intent-filter>
        <action android:name="AAA"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

- 보호된 컴포넌트를 이용하는 앱

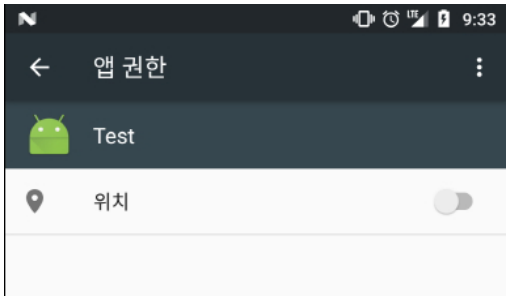
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.test">

    <uses-permission android:name="com.test.permission.SOME_PERMISSION"/>
    <!--중략-->
</manifest>
```

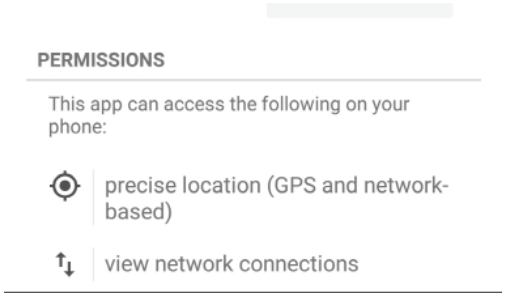
protectionLevel 속성

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

- android.permission.ACCESS_NETWORK_STATE는 “normal”로 선언,
  android.permission.ACCESS_FINE_ LOCATION은 “dangerous”로 선언.

9.1 퍼미션





- 5.1까지는 권한 화면에서 어떤 퍼미션이 사용되는지 정보 성격으로만 알려주지만, 6.0 이상부터는 사용자에게 해당 권 한을 부여할 것인지 선택

9.1 퍼미션

시스템 퍼미션

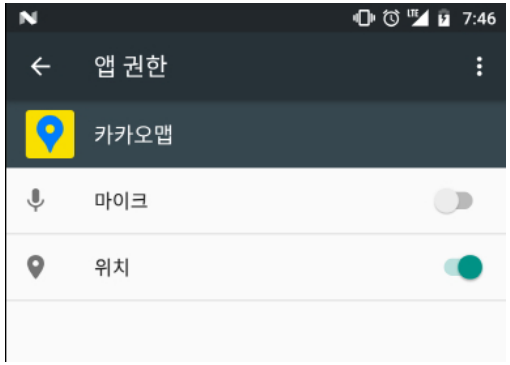
퍼미션 그룹	퍼미션
CALENDAR	READ_CALENDAR
	WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS
	WRITE_CONTACTS
	GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION
	ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE
	CALL_PHONE
	READ_CALL_LOG
	WRITE_CALL_LOG
	ADD_VOICEMAIL
	USE_SIP
	PROCESS_OUTGOING_CALLS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

9.1 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

9.1 퍼미션

9.1.2. 안드로이드 6.0 (API Level 23) 변경사항



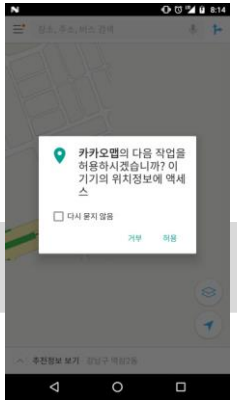
- 6.0 이전까지 퍼미션은 일종의 개발자 신고제
- 6.0부터는 퍼미션을 사용자가 거부 가능

- 퍼미션 상태 를 확인
- int checkSelfPermission (Context context, String permission)
- PERMISSION_GRANTED: 퍼미션이 부여된 상태
- PERMISSION_DENIED: 펴미션이 부여되지 않은 상태

```
if(ContextCompat.checkSelfPermission(this,
    Manifest.permission.READ_EXTERNAL_STORAGE) ==
    PackageManager.PERMISSION_GRANTED){
    //...
}
```

9.1 퍼미션

- 퍼미션 허용을 요청
- void requestPermissions(Activity activity, String[] permissions, int requestCode)




```
ActivityCompat.requestPermissions(this,
new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},200 );
```

- 퍼미션을 허용했는지 판단
- void onRequestPermissionsResult(int requestCode, String[] permissions,
  int[] grantResults)

```
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    if(requestCode==200 && grantResults.length>0) {
        if(grantResults[0]==PackageManager.PERMISSION_GRANTED)
            //.................
        if(grantResults[1]==PackageManager.PERMISSION_GRANTED)
            //.................
    }
}
```

9.2 파일에 읽고 쓰기

파일 관련 프 로그램은 대부분 자바 API를 그대로 사용

- File: 파일 및 디렉터리를 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

Environment

- Environment.getExternalStorageState(): 외부 저장 공간 상태
- Environment.getExternalStorageDirectory().getAbsolutePath(): 외부 저장 공간 경로
- Environment.getDataDirectory().getAbsolutePath(): 내부 저장 공간 경로

9.2 파일에 읽고 쓰기

9.2.1. 외부 저장 공간 이용

- 외부 저장 공간을 제공하는지 판단

```
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;
}
```

- 퍼미션

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.test3_9">
    <uses-permission android:name=
                "android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name=
                "android.permission.READ_EXTERNAL_STORAGE" />
    <!-- 중략 -->
</manifest>
}
```

9.2 파일에 읽고 쓰기

- 문자열 데이터를 저장

```
FileWriter writer;
try {
    //공간 저장 외부 root 하    에위myApp획득 경로 폴더 이    는라
    String dirPath = Environment.getExternalStorageDirectory()
            .getAbsolutePath() + "/myApp";
    File dir = new File(dirPath);
    //만 새로 없    면더폴    가듬다
    if(!dir.exists()){
        dir.mkdir();
    }
    //myApp 밑어 폴더 myfile.txt 지정 파일
    File file=new File(dir+"/myfile.txt");
    //만 새로 없    면때파    이을다
    if(!file.exists()){
        file.createNewFile();
    }
    //파일 쓰기
    writer = new FileWriter(file, true);
    writer.write(content);
    writer.flush();
    writer.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

9.2 파일에 읽고 쓰기

- createTempFile ( ) 함수이용

```
File tempFile = File.createTempFile("IMG", ".jpg",dir);
```

- 공용 폴더를 사용

```
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

9.2 파일에 읽고 쓰기

- 파일을 읽기

```
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);
   }
   reader.close();
}catch (Exception e){
   e.printStackTrace();
}
```

9.2.2. 내부 저장 공간 이용

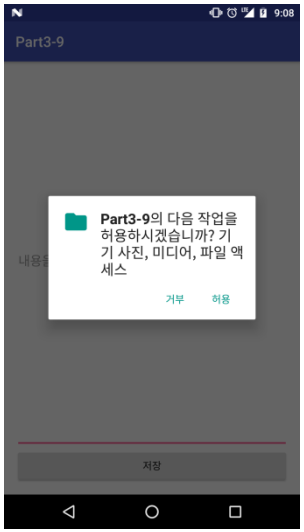
- 앱의 데이터를 내부 저장 공간에 저장하려면 getFileDir ( ) 함 수로 경로 획득

Step by Step 9-1 – 파일 다루기

File에 데이터를 Read/Write 하는 실습

•간단한 메모장 기능을 가정해서 작성
•유저가 입력한 글을 파일로 저장하고 저장된 파일의 내용을 읽어서 화면에 출력

1. Module 생성
2. 퍼미션 부여
3. 결과 확인을 위한 Activity 생성
4. activity_read_file.xml 작성
5. ReadFileActivity 작성
6. activity_main.xml 복사
7. MainActivity 작성
8. 실행



9.3 SharedPreferences와 앱 설정 자동화

9.3.1. SharedPreferences

- 데이터를 간단하게 키-값(key-value) 성격으로 저장
- 파일(XML)로 저장

SharedPreference 객체를 하나 획 득
- getPreferences(int mode)
- getSharedPreferences(String name, int mode)
- PreferenceManager.getDefaultSharedPreferences(Context context)

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

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

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

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

9.3 SharedPreferences와 앱 설정 자동화

데이터를 저장하려면 Editor 클래스의 함수를 이용

- 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)

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

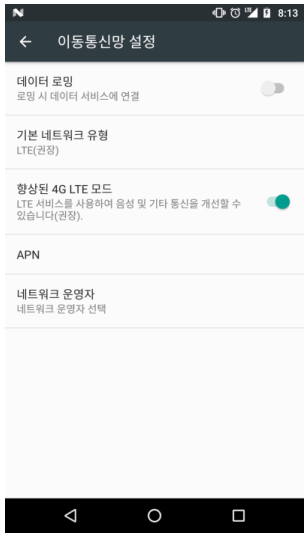
데이터를 획득

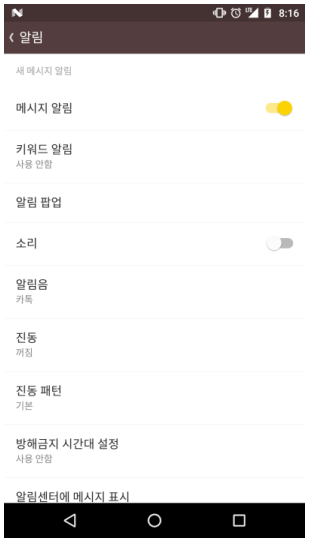
- getBoolean(String key, boolean defValue)
- getFloat(String key, float defValue)
- getInt(String key, int defValue)
- getLong(String key, long defValue)
- getString(String key, String defValue)

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

9.3 SharedPreferences와 앱 설정 자동화

9.3.2. 앱 설정 자동화





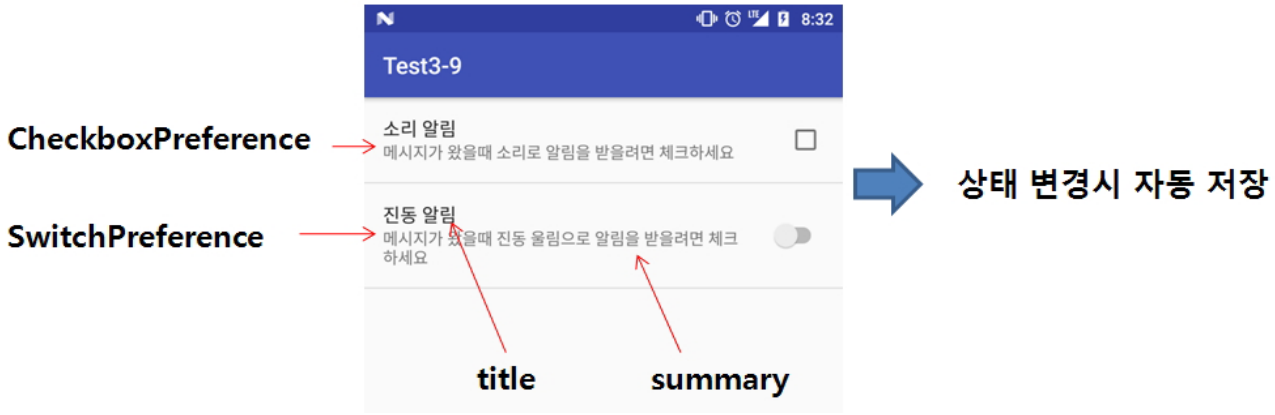
- 설정 화면을 위한 XML 파일을 하나 준비
- res 하위에 “xml”이라는 폴더에 XML 파일 작성

- <PreferenceScreen>: 설정 화면 단위.
- <PreferenceCategory>: 설정 여러 개를 시각적으로 묶어서 표현
- <CheckboxPreference>: 체크박스가 나오는 설정
- <EditTextPreference>: 글 입력을 위한 설정
- <ListPreference>: 항목 다이얼로그를 위한 설정
- <MultiSelectListPreference>: 항목 다이얼로그인데 체크박스가 자동 추가
- <RingtonPreference>: Rington 선택을 위한 설정
- <SwitchPreference>: Switch를 이용한 설정

9.3 SharedPreferences와 앱 설정 자동화

<CheckboxPreference>, <SwitchPreference>

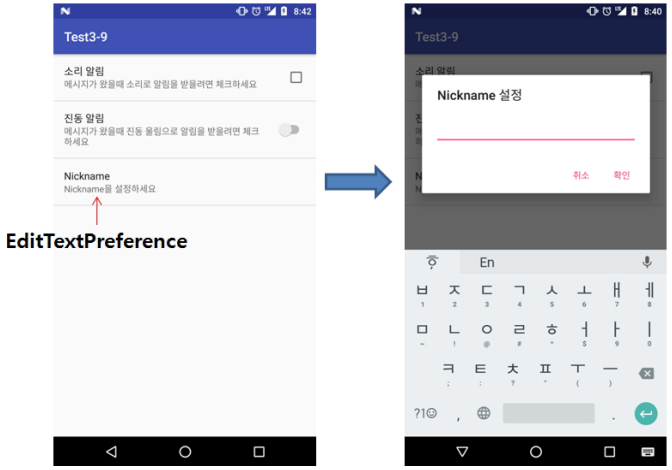
```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
   <CheckBoxPreference
      android:key="message"
      android:title="알림 소리"
      android:summary="받 알    을랄    로픽체          오세려면"/>
   <SwitchPreference
      android:key="vibrate"
      android:title="알림 진동"
      android:summary=" 받 알    을랄       로 진동으체          오세려면"/>
      />
</PreferenceScreen>
```



9.3 SharedPreferences와 앱 설정 자동화

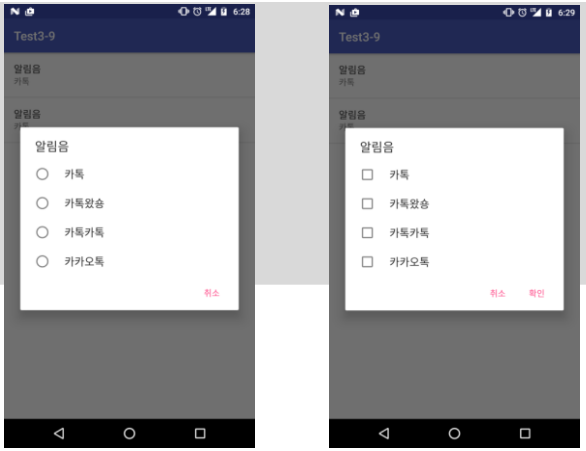
<EditTextPreference>

```
<EditTextPreference
    android:key="nickname"
    android:title="Nickname"
    android:summary="Nickname설
    android:dialogTitle="Nickname 설정"
    />
```



<ListPreference>, <MultiSelectListPreference>

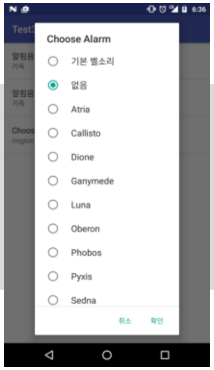
```
<ListPreference
    android:key="sound"
    android:title="알    음림
    android:summary="카톡"
    android:entries="@array/array_voice"
    android:entryValues="@array/array_voice"
    />
```



9.3 SharedPreferences와 앱 설정 자동화

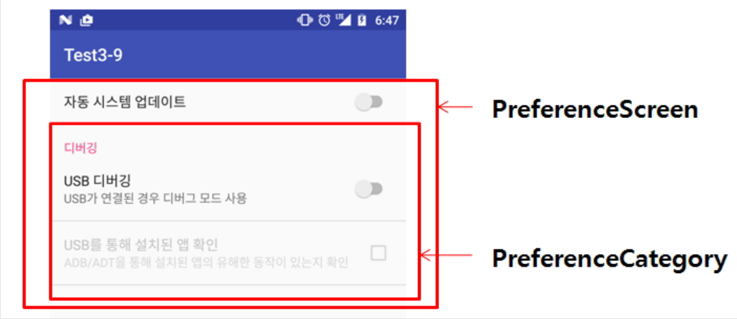
<RingtonePreference>

```
<RingtonePreference
   android:title="Choose Alarm"
   android:key="ringtone"
   android:summary="rington"
   />
```



<PreferenceCategory>

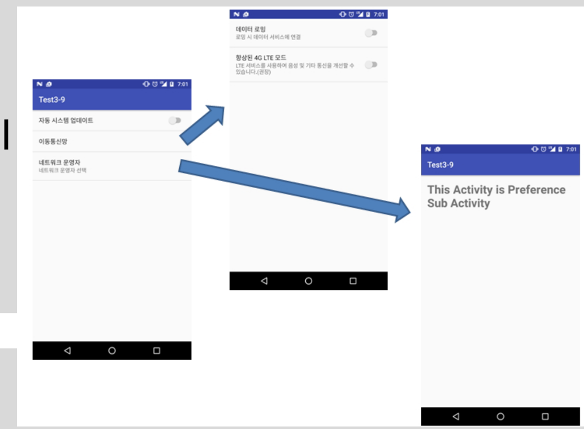
```
<PreferenceCategory android:title="디버깅">
   <SwitchPreference
      android:defaultValue="false"
      android:key="debugging"
      android:summary="USB사용 모드 디버그 경우 연결된 "
      android:title="USB 디버깅"/>

   <CheckBoxPreference
      android:defaultValue="false"
      android:dependency="debugging"
      android:key="usb_app"
      android:summary="ADB/ADT확인 있는지 동작이 유해한 앱의 설치된 통해 을"
      android:title="USB확인 앱 설치된 통해 를" />
</PreferenceCategory>
```



9.3 SharedPreferences와 앱 설정 자동화

<PreferenceScreen>

```
<PreferenceScreen android:title="이        망통">
  <SwitchPreference
    android:defaultValue="false"
    android:key="roaming"
    android:summary="연결 서      에비데   터에 로밍"
    android:title="로밍 데   터이/>

  <SwitchPreference
    android:key="lte_mode"
    android:summary="LTE 수 개   할통   을선터타 및 음성 사      여용       를비
                  있      다스(권장)"
    android:title="향   된상4G LTE 모드" />

</PreferenceScreen>
```



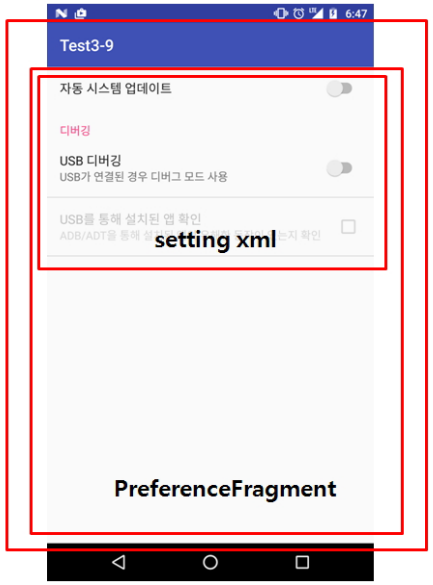

```
<PreferenceScreen
  android:title="운   자영      크워트
  android:summary="선택 운   자영      크워트>
  <intent android:targetPackage="com.example.test3_9"
    android:targetClass="com.example.test3_9.SettingsSubActivity"/>
</PreferenceScreen>
```

9.3 SharedPreferences와 앱 설정 자동화

9.3.3. 앱 설정 자동화 적용

PreferenceFragment

```
public class SettingPreferenceFragment extends PreferenceFragment {

    @Override
    public void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.settings_preference);
    }
}
```

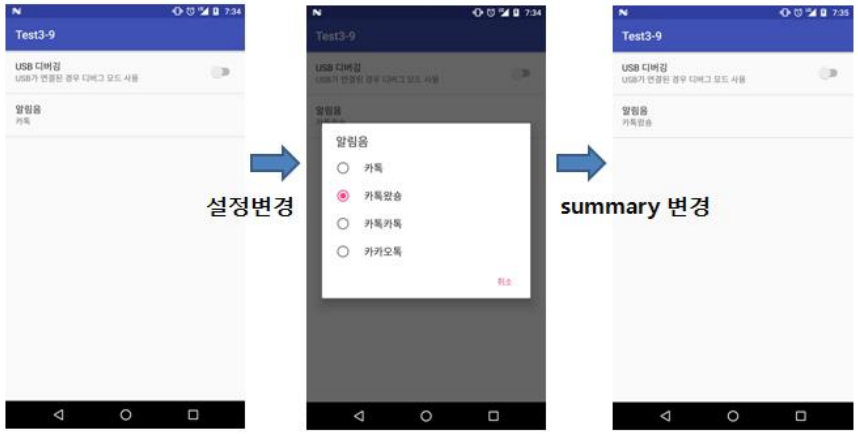


- 액티비티의 레이아웃 XML 파일에서 <fragment> 태그를 이용하여 PreferenceFragment 서브 클래스를 등록

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/settings_fragment"
    android:name="com.example.test3_9.SettingPreferenceFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

9.3 SharedPreferences와 앱 설정 자동화

summary 변경




```
prefs = PreferenceManager.getDefaultSharedPreferences(getActivity());

if (!prefs.getString("sound", "").equals(""))
    soundPreference.setSummary(prefs.getString("sound", "카톡"));
```

Preference 이벤트 처리

```
refs.registerOnSharedPreferenceChangeListener(prefListener);
```

9.3 SharedPreferences와 앱 설정 자동화

```
SharedPreferences.OnSharedPreferenceChangeListener prefListener =
    new SharedPreferences.OnSharedPreferenceChangeListener() {

    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                          String key) {
        if (key.equals("sound")) {
            soundPreference.setSummary(prefs.getString("sound", "카톡"));
        }
    }
};
```

Step by Step 9-2 – 설정 자동화

설정 자동화 부분의 테스트

•카카오톡의 설정중 알림 설정 부분 화면의 일부분을 동일하게 만드는 것을 목적으로 진행

1. Activity 생성
2. 문자열 배열 등록
3. settings_preference.xml 작성
4. SettingPreferenceFragment.java 작성
5. activity_lab9_2.xml 작성
6. Lab9_2Activity.java 실행