

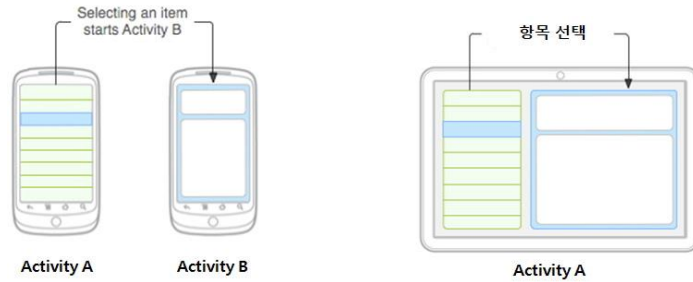


17장. Support 라이브러리 활용

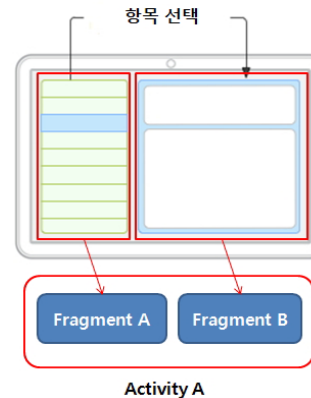
17.1 Fragment

17.1.1. Fragment 이해

- Fragment는 API Level 11(Android 3.0)에서 추가된 뷰



- 액티비티처럼 이용할 수 있는 뷰
- 태블릿 PC용 앱의 화면을 작성할 때 액티비티가 복잡하게 작성되는 문제를 해결하는 목적으로 Fragment를 제공



17.1 Fragment

17.1.2. Fragment 작성법

- Fragment의 화면을 위한 레이아웃 XML 파일

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="One Fragment!!"
        android:textStyle="bold"
        android:textSize="30dp"
        android:gravity="center"
    />
</LinearLayout>
```

- Fragment 클래스

```
public class OneFragment extends Fragment {
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_one, container, false);
    }
}
```

17.1 Fragment

- 액티비티의 레이아웃 XML 파일에서 Fragment는 <fragment> 태그로 등록

```
<fragment
    android:id="@+id/fragment_one"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    class="com.example.test6_17.OneFragment"
/>
```

- 자바 코드에서 동적으로 액티비티 화면에 Fragment를 출력
- Fragment를 포함할 LinearLayout을 준비

```
<LinearLayout
    android:id="@+id/main_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

</LinearLayout>
```

17.1 Fragment

- FragmentManager 클래스로 FragmentTransaction 클래스를 획득

```
oneFragment=new OneFragment();
```

```
FragmentTransaction ft=manager.beginTransaction();  
ft.add(R.id.main_container, oneFragment);  
ft.commit();
```

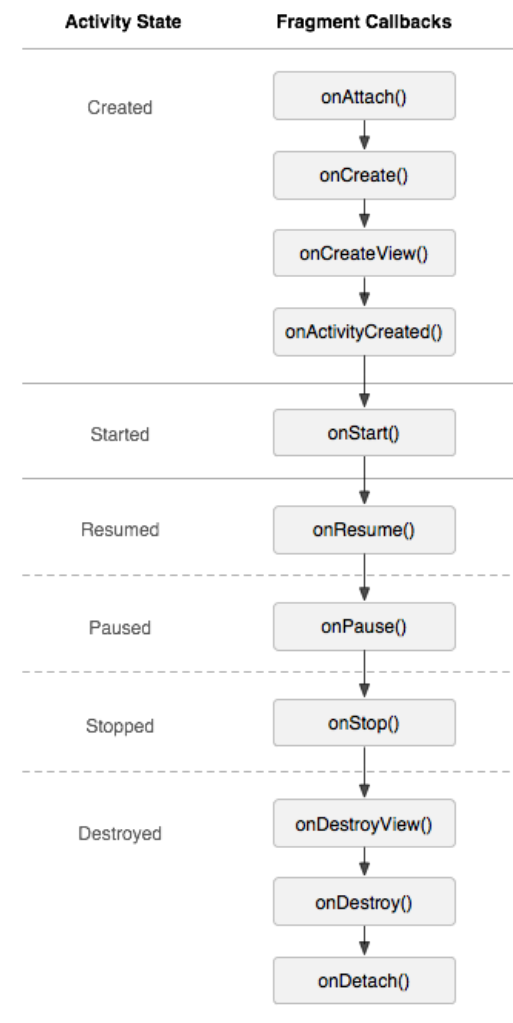
- add(int containerViewId, Fragment fragment): 새로운 Fragment를 화면에 추가. id 영역에 추가
- add(int containerViewId, Fragment fragment, String tag): id 영역에 Fragment 추가하며 추가한 Fragment의 구분자를 태그명으로 설정
- replace(int containerViewId, Fragment fragment): id 영역에 추가된 Fragment를 대체
- replace(int containerViewId, Fragment fragment, String tag): id 영역에 추가된 Fragment를 대체하면서 tag 이름 설정
- remove(Fragment fragment): 추가된 Fragment 제거
- commit(): 화면 적용

17.1 Fragment

17.1.3. Fragment 생명주기

- Fragment의 생명주기는 액티비티 생명주기와 동일하며 Fragment만을 위한 생명주기 함수가 더 추가된 구조
- BackStack은 Fragment가 화면에 안 보이게 되는 순간 제거하지 않고 저장했다가 다시 이용
- Fragment를 BackStack에 추가

```
ft.addToBackStack(null);
```

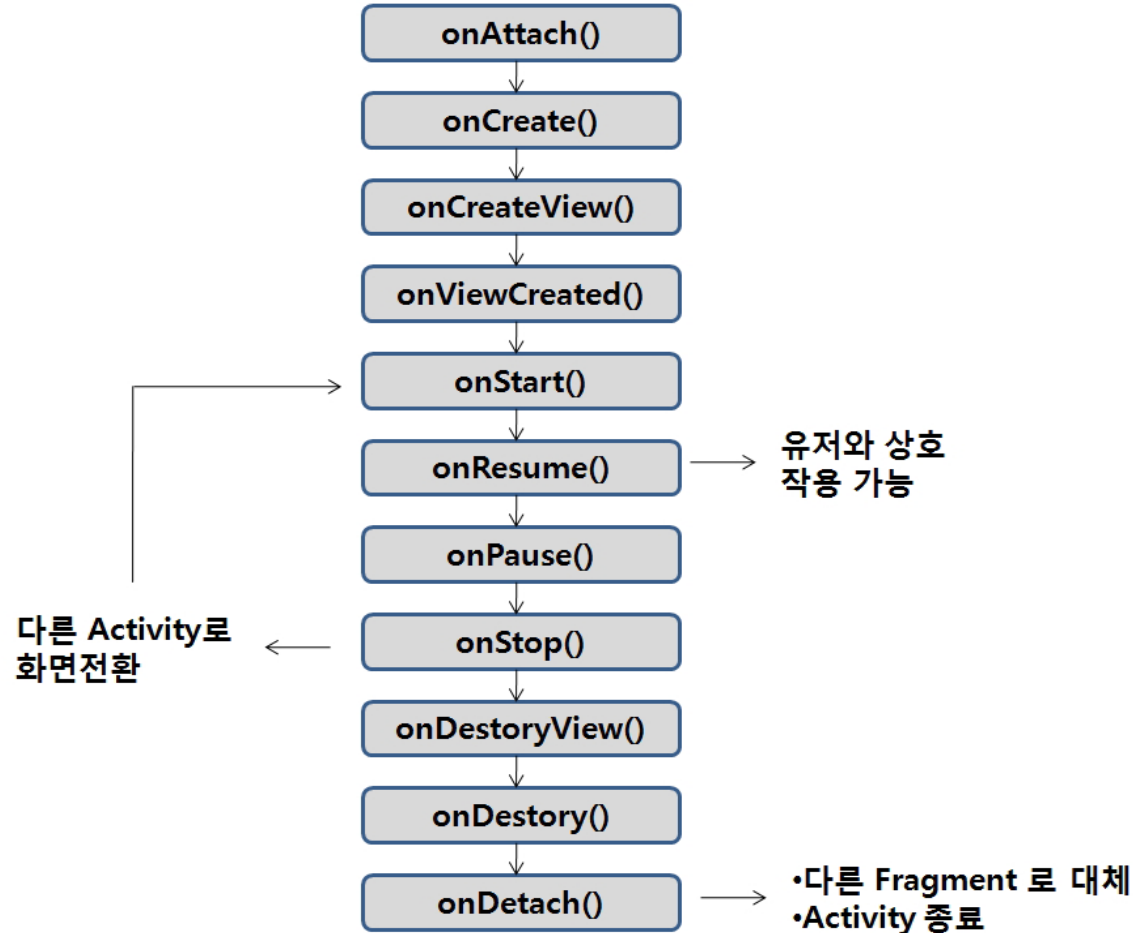


17.1 Fragment

- `onAttach(Activity activity)`: Fragment가 액티비티에 포함되는 순간 호출
- `onCreate(Bundle savedInstanceState)`: 액티비티의 `onCreate()` 함수와 동일
- `onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)`:
Fragment의 UI 구성을 위해 호출. 이곳에서 반환하는 뷰가 Fragment 화면에 출력
- `onActivityCreated(Bundle savedInstanceState)`: Fragment의 액티비티 생성이 완료된 순간 호출
- `onStart()`: 액티비티의 `onStart()` 함수와 동일
- `onResume()`: 액티비티의 `onResume()` 함수와 동일. 사용자 화면에 출력되면서 사용자와 상호 작용
이 가능한 상태
- `onPause()`: 액티비티의 `onPause()` 함수와 동일
- `onStop()`: 액티비티의 `onStop()` 함수와 동일
- `onDestroyView()`: Fragment가 화면에서 사라진 후 BackStack에 추가된 후 호출
- `onDestroy()`: 액티비티의 `onDestroy()` 함수와 동일
- `onDetach()`: Fragment가 액티비티에서 제거될 때 호출

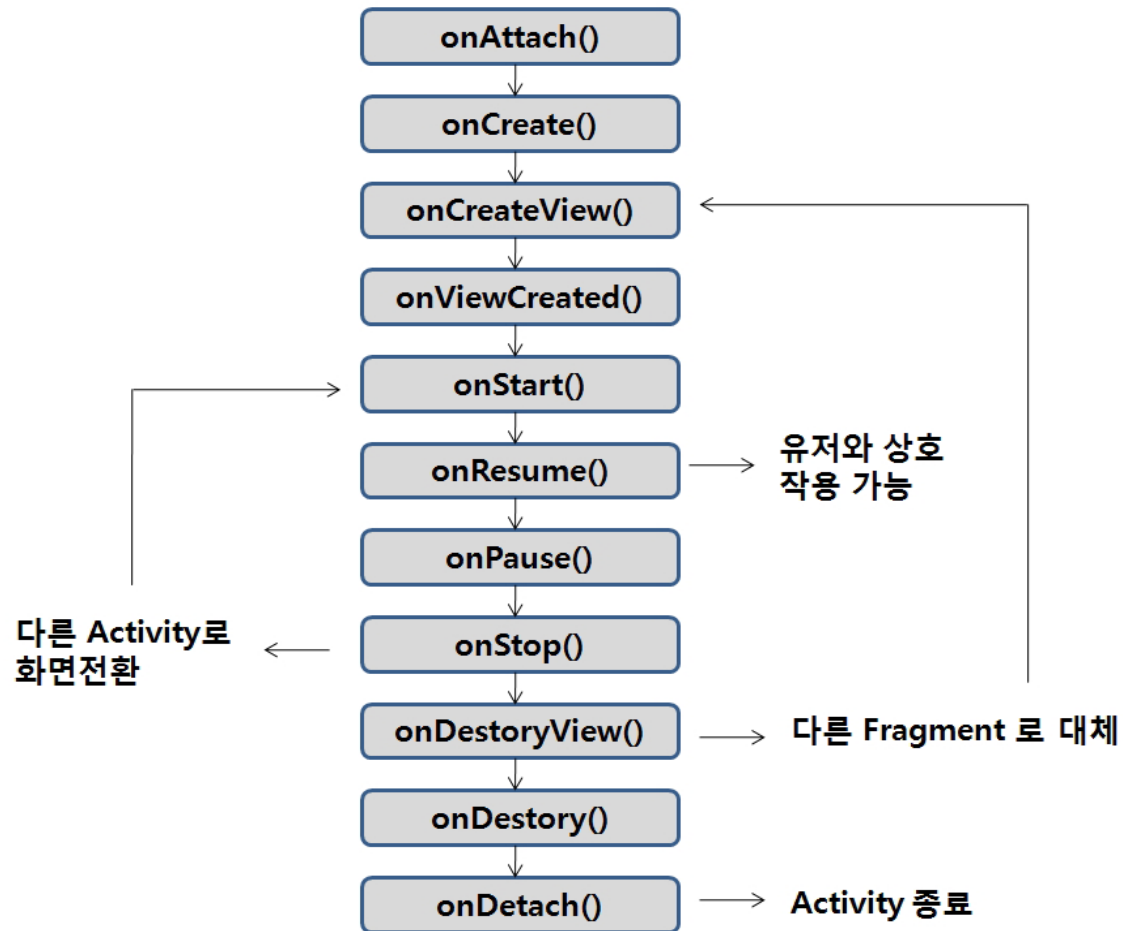
17.1 Fragment

- BackStack을 사용하지 않을 때 Fragment의 생명주기



17.1 Fragment

- BackStack을 사용할 때



17.1 Fragment

17.1.4. 다양한 Fragment

- ListFragment
- ListFragment는 ListView로 화면을 구성할 때 ListView와 관련된 내용만 액티비티에서 분리 해서 Fragment로 구현할 수 있게 만든 클래스

```
public class OneFragment extends ListFragment {  
    @Override  
    public void onViewCreated(View view, Bundle savedInstanceState) {  
        super.onViewCreated(view, savedInstanceState);  
        String[] datas={"박찬호","류현진","김현수","오승환"};  
        ArrayAdapter<String> aa=new ArrayAdapter<String>(getActivity(),  
            android.R.layout.simple_list_item_1,datas);  
        setListAdapter(aa);  
    }  
}
```



- 항목 선택 이벤트

```
public void onListItemClick(ListView l, View v, int position, long id) {  
    super.onListItemClick(l, v, position, id);  
    //...  
}
```

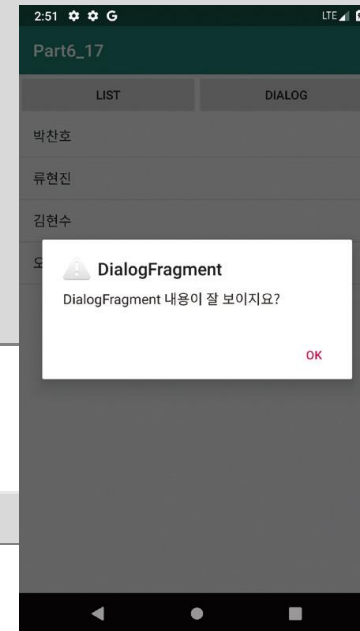
17.1 Fragment

- DialogFragment

```
public class ThreeFragment extends DialogFragment {  
    @Override  
    public Dialog onCreateDialog(Bundle savedInstanceState) {  
        AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());  
        builder.setIcon(android.R.drawable.ic_dialog_alert);  
        builder.setTitle("DialogFragment");  
        builder.setMessage("DialogFragment 내용이 잘 보이지요?");  
        builder.setPositiveButton("OK", null);  
        AlertDialog dialog=builder.create();  
        return dialog;  
    }  
}
```

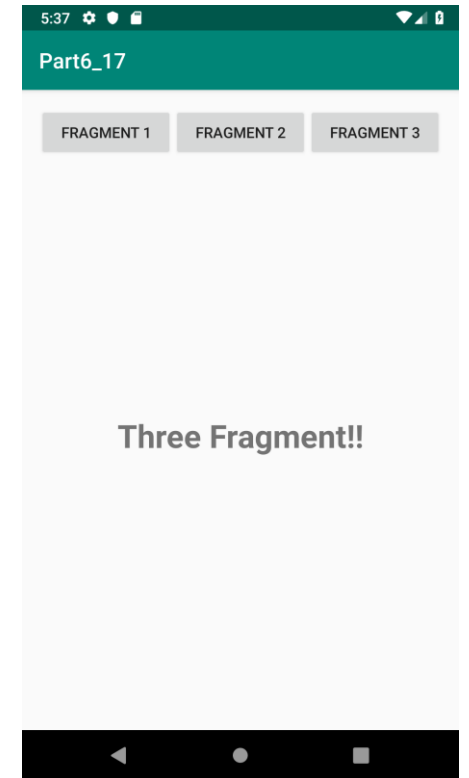
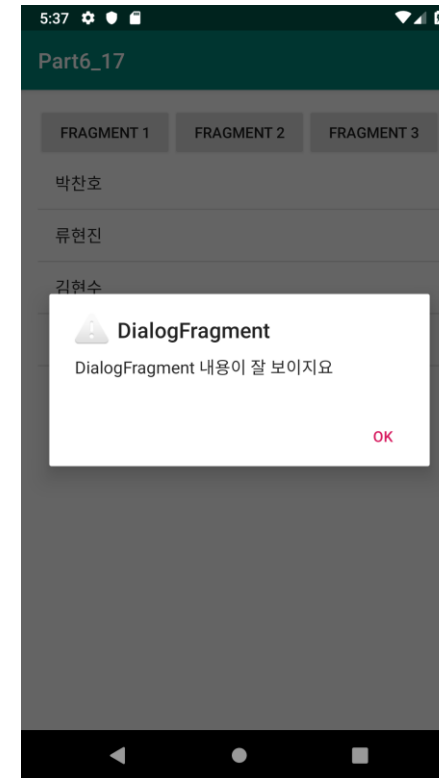
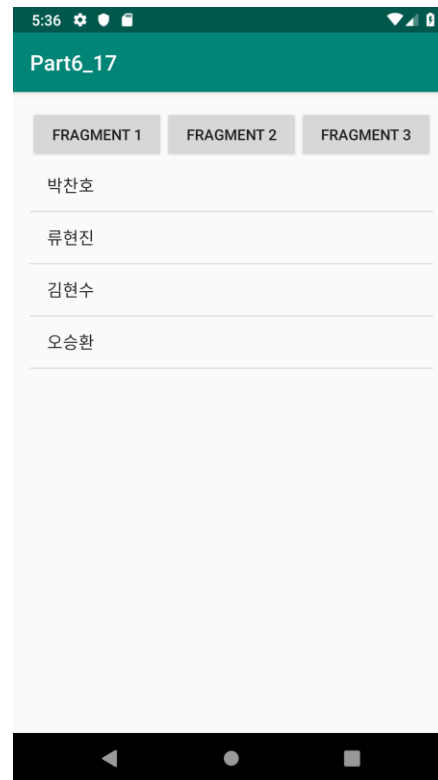
- DialogFragment 클래스의 show() 함수를 이용하여 출력

```
threeFragment.show(manager, null);
```



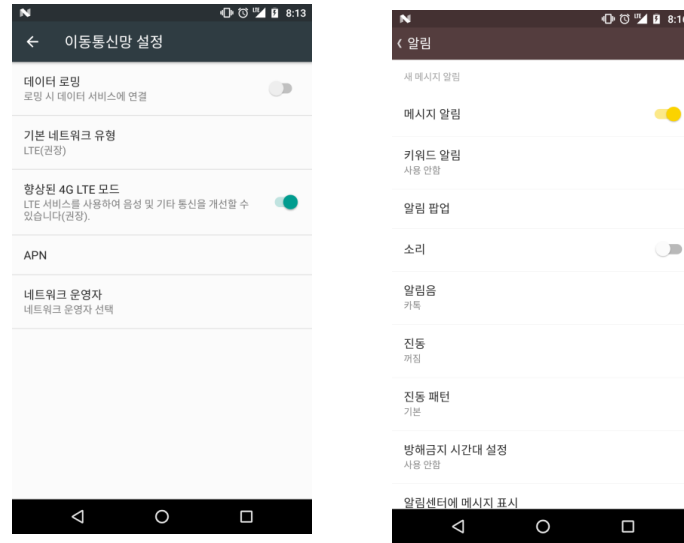
Step by Step 17-1 - Fragment

- 모듈 생성
- 파일 복사
- OneFragment 작성
- TwoFragment 작성
- ThreeFragment 작성
- MainActivity 작성
- 실행



17.2 PreferenceFragmentCompat을 이용한 앱 설정

17.2.1. 앱 설정 자동화

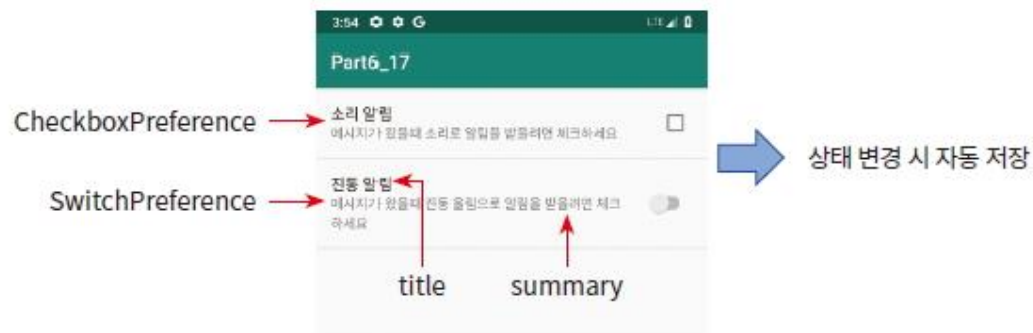


- 설정 화면을 위한 XML 파일을 하나 준비
- res 하위에 “xml”이라는 폴더에 XML 파일 작성
- <PreferenceScreen>: 설정 화면 단위.
- <PreferenceCategory>: 설정 여러 개를 시각적으로 묶어서 표현
- <CheckBoxPreference>: 체크박스가 나오는 설정
- <EditTextPreference>: 글 입력을 위한 설정
- <ListPreference>: 항목 다이얼로그를 위한 설정
- <MultiSelectListPreference>: 항목 다이얼로그인데 체크박스가 자동 추가
- <RingtonePreference>: Ringtone 선택을 위한 설정
- <SwitchPreference>: Switch를 이용한 설정

17.2 PreferenceFragmentCompat을 이용한 앱 설정

<CheckBoxPreference>, <SwitchPreference>

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
  <CheckBoxPreference
    android:key="message"
    android:title="소      알림"
    android:summary="소      알림을 받을까요?"
  />
  <SwitchPreference
    android:key="vibrate"
    android:title="진      동"
    android:summary="진      동을 받을까요?"
  />
</PreferenceScreen>
```

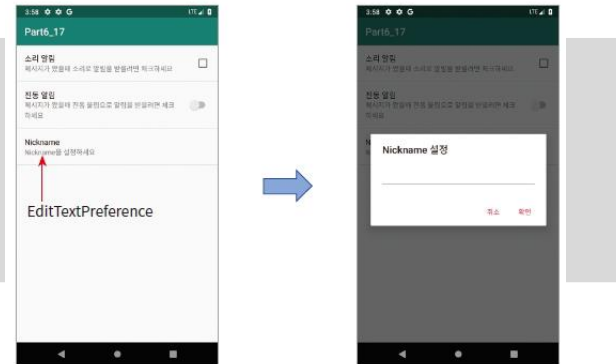


17.2 PreferenceFragmentCompat을 이용한 앱 설정

<EditTextPreference>

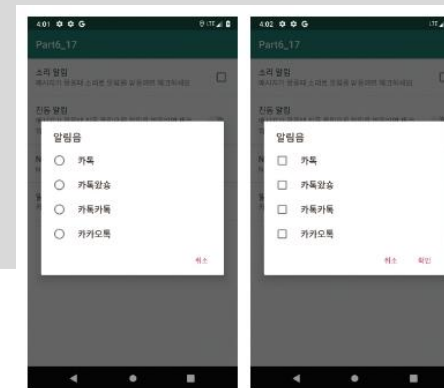
```
<EditTextPreference
    android:key="nickname"
    android:title="Nickname"
    android:summary="Nickname을
    android:dialogTitle="Nickname 설정"
/>
```

설정



<ListPreference>, <MultiSelectListPreference>

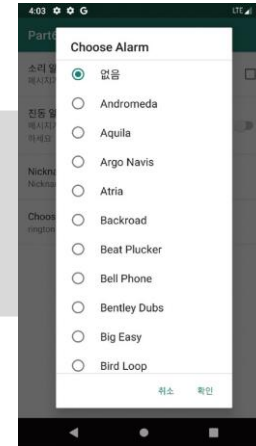
```
<ListPreference
    android:key="sound"
    android:title="알림음"
    android:summary="카톡"
    android:entries="@array/array_voice"
    android:entryValues="@array/array_voice"
/>
```



17.2 PreferenceFragmentManager을 이용한 앱 설정

<RingtonePreference>

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



<PreferenceCategory>

```
<PreferenceCategory android:title="디깅"
    <SwitchPreference
        android:defaultValue="false"
        android:key="debugging"
        android:summary="USB가
        android:title="USB 디깅"/>

    <CheckBoxPreference
        android:defaultValue="false"
        android:dependency="debugging"
        android:key="usb_app"
        android:summary="ADB/ADT을
        android:title="USB를
    </PreferenceCategory>
```



17.2 PreferenceFragmentCompat을 이용한 앱 설정

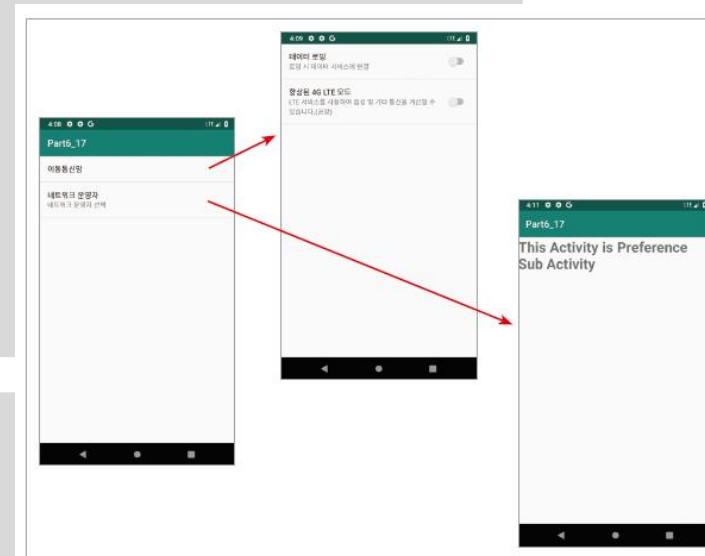
<PreferenceScreen>

```
<PreferenceScreen android:title="이 망상동">  
  <SwitchPreference  
    android:defaultValue="false"  
    android:key="roaming"  
    android:summary="로  
    android:title="데 멍미"/>
```

```
  <SwitchPreference  
    android:key="lte_mode"  
    android:summary="LTE 서  
    android:title="향 된상G LTE 모드" />
```

</PreferenceScreen>

```
<PreferenceScreen  
  android:title="네 정취"  
  android:summary="네 신상권장"  
  <intent android:targetPackage="com.example.test3_9"  
    android:targetClass="com.example.test3_9.SettingsSubActivity"/>  
</PreferenceScreen>
```



17.2 PreferenceFragmentManager을 이용한 앱 설정

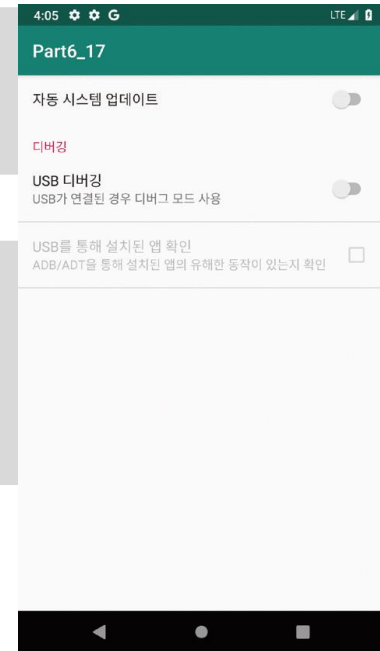
17.2.2. PreferenceFragmentManager

```
dependencies {  
    ...  
    implementation 'com.android.support:preference-v14:28.0.0'  
}
```

```
public class SettingPreferenceFragment extends PreferenceFragmentManager {  
    @Override  
    public void onCreatePreferences(Bundle bundle, String s) {  
        setPreferencesFromResource(R.xml.settings_preference, s);  
    }  
}
```

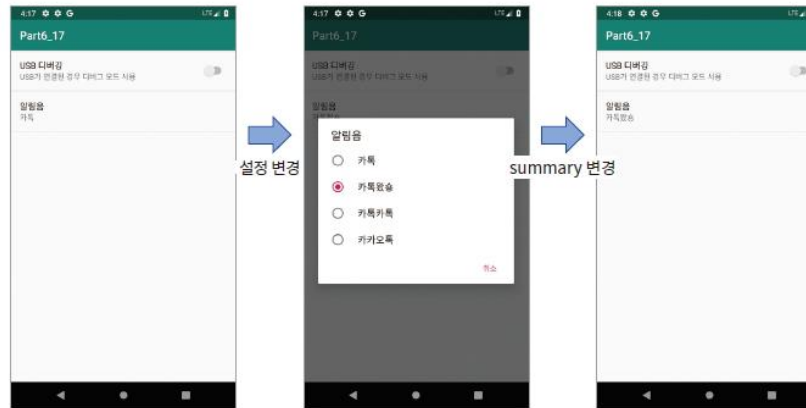
- 액티비티의 레이아웃 XML 파일에서 <fragment> 태그를 이용하여 PreferenceFragmentManager 서브 클래스를 등록

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/settings_fragment"  
    android:name="com.example.test6_17.SettingPreferenceFragment"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
/>
```



17.2 PreferenceFragmentCompat을 이용한 앱 설정

summary 변경



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

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

Preference 이벤트 처리

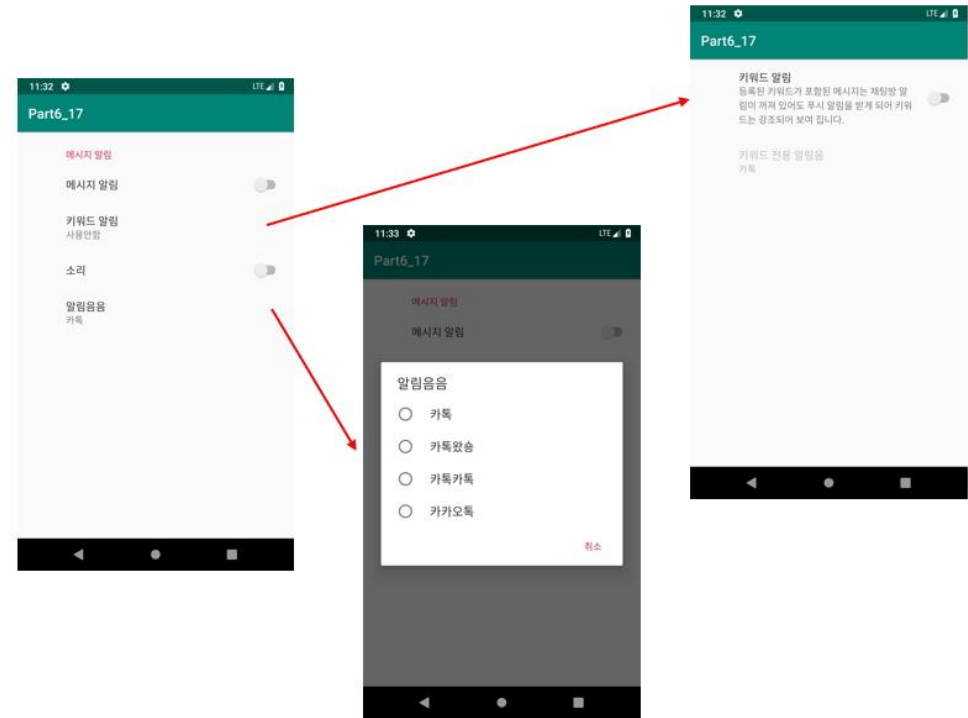
```
refs.registerOnSharedPreferenceChangeListener(prefListener);
```

17.2 PreferenceFragmentCompat을 이용한 앱 설정

```
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 17-2 – 설정 자동화

- 액티비티 생성
- 라이브러리 등록
- 문자열 배열 등록
- settings_preference.xml 작성
- settingPreferenceFragment.java 작성
- sctivity_lab17_2.xml
- Lab17_2Activity.java



17.3 ViewPager

- 사용자 손가락을 따라가며 순서대로 좌우 화면이 슬라이드되어 나타나는 구성
- support v4에서 제공
- XML 파일에 등록

```
<android.support.v4.view.ViewPager
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- Adapter를 만들어 적용



17.3 ViewPager

- PagerAdapter와 항목(화면)을 Fragment로 개발하기 위한 FragmentPagerAdapter, 두 개가 제공

```
class MyPagerAdapter extends PagerAdapter {
    @Override
    public int getCount() {}
    @Override
    public boolean isViewFromObject(View arg0, Object arg1) {}
    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        if(position==0){

            TextView tv=new TextView(Lab2Activity.this);
            //... container.addView(tv,position);
            return tv;
        } //... return null;
    }
    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        container.removeView((View)object);
    }
}
```

- getCount(): 항목 개수 결정
- instantiateItem(ViewGroup container, int position): 항목 구성
- isViewFromObject(View arg0, Object arg1): 항목을 위한 뷰 결정
- destroyItem(ViewGroup container, int position, Object object): 뷰 소멸

17.3 ViewPager

- FragmentPagerAdapter를 상속받아 작성

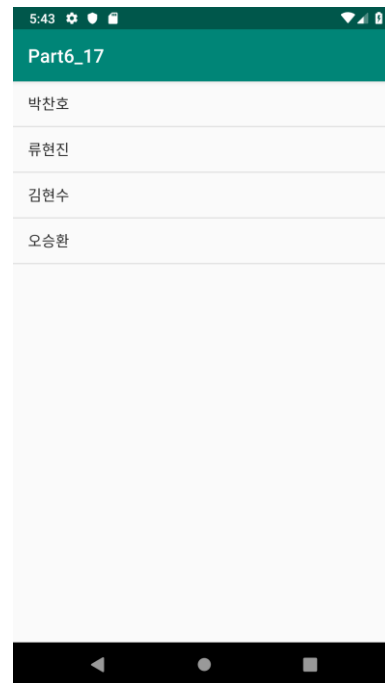
```
class MyPagerAdapter extends FragmentPagerAdapter {  
    ArrayList<Fragment> fragments;  
    public MyPagerAdapter(FragmentManager manager){  
        super(manager);  
        //...  
    }  
    @Override  
    public int getCount() { //... }  
    @Override  
    public Fragment getItem(int position) {  
        //...  
    }  
}
```

- Adapter를 ViewPager에 적용

```
MyPagerAdapter pagerAdapter=new MyPagerAdapter(getSupportFragmentManager());  
pager.setAdapter(pagerAdapter);
```

Step by Step 17-3 - ViewPager

- 액티비티 생성
- activity_lab17_3.xml 작성
- Lab17_3Activity 작성
- 실행



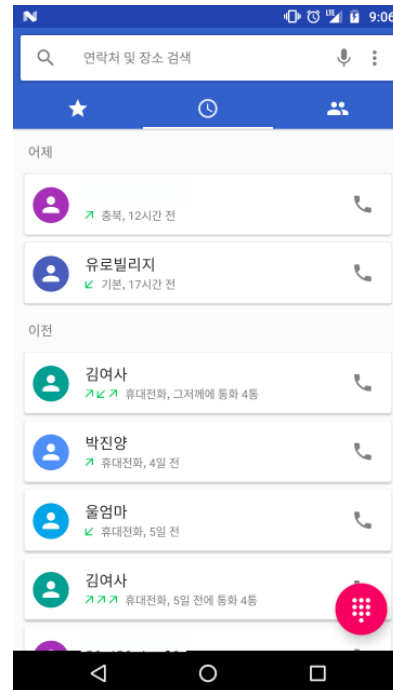
17.4 RecyclerView

17.4.1. RecyclerView 소개

- RecyclerView는 API Level 21(Android 5.0)이 나오면서 support:recyclerView-v7 라이브러리로 제공된 클래스

```
implementation 'com.android.support:recyclerview-v7:28.0.0'
```

- RecyclerView는 API Level 21(Android 5.0)이 나오면서 support:recyclerView-v7 라이브러리로 제공된 클래스



17.4 RecyclerView



- Adapter: RecyclerView 항목 구성
- ViewHolder: 각 항목 구성 뷰의 재활용을 목적으로 View Holder 역할
- LayoutManager : 항목의 배치
- ItemDecoration: 항목 꾸미기
- ItemAnimation: 아이템이 추가, 제거, 정렬될 때의 애니메이션 처리

17.4.2. Adapter, ViewHolder

- RecyclerView를 하나 준비

```
<android.support.v7.widget.RecyclerView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/lab3_recycler"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"/>
```

17.4 RecyclerView

- ViewHolder 클래스

```
private class MyViewHolder extends RecyclerView.ViewHolder {  
    public TextView title;  
    public MyViewHolder(View itemView) {  
        super(itemView);  
        title = (TextView) itemView.findViewById(android.R.id.text1);  
    }  
}
```

- Adapter

```
private class MyAdapter extends RecyclerView.Adapter<MyViewHolder> {  
    private List<String> list;  
    public MyAdapter(List<String> list) {  
        this.list = list;  
    }  
    public MyViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {  
        View view = LayoutInflater.from(viewGroup.getContext())  
            .inflate(android.R.layout.simple_list_item_1, viewGroup, false);  
        return new MyViewHolder(view);  
    }  
    public void onBindViewHolder(MyViewHolder viewHolder, int position) {  
        String text = list.get(position);  
        viewHolder.title.setText(text);  
    }  
    public int getItemCount() {  
        return list.size();  
    }  
}
```

17.4 RecyclerView

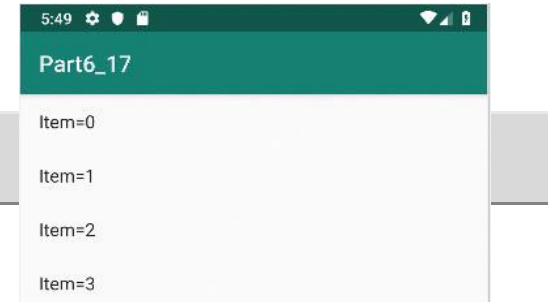
```
private class MyAdapter extends RecyclerView.Adapter<MyViewHolder> {  
    //.....  
    layout inflate ① public MyViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {  
        View view = LayoutInflater.from(viewGroup.getContext())  
        ViewHolder 리턴 ④ .inflate(android.R.layout.simple_list_item_1, viewGroup, false);  
        return new MyViewHolder(view);  
    }  
    @Override  
    항목 구성 ⑤ public void onBindViewHolder(MyViewHolder viewHolder, int position) {  
        String text = list.get(position);  
        viewHolder.title.setText(text);  
    } ⑥ 전달된 ViewHolder 이용 ② Layout 초기화후 ViewHolder 생성  
}  
private class MyViewHolder extends RecyclerView.ViewHolder {  
    public TextView title;  
    public MyViewHolder(View itemView) {  
        super(itemView);  
        title = (TextView) itemView.findViewById(android.R.id.text1);  
    }  
}
```

③ 필요 View findViewById

17.4 RecyclerView

- Adapter를 RecyclerView에 적용

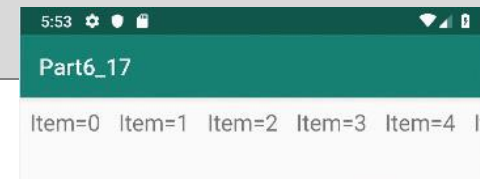
```
recyclerView.setLayoutManager(new LinearLayoutManager(this));  
recyclerView.setAdapter(new MyAdapter(list));
```



17.4.3. LayoutManager

- 항목을 어떻게 배치
- LinearLayoutManager: 수평, 수직으로 배치
- GridLayoutManager: 그리드 화면으로 배치
- StaggeredGridLayoutManager: 높이가 불규칙한 그리드 화면으로 배치
- LinearLayoutManager

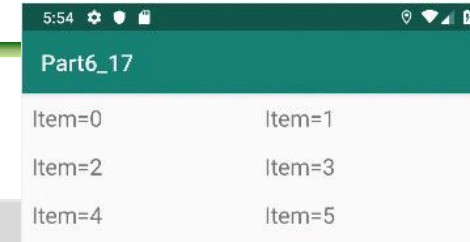
```
LinearLayoutManager linearManager = new LinearLayoutManager(this);  
linearManager.setOrientation(LinearLayoutManager.HORIZONTAL);  
recyclerView.setLayoutManager(linearManager);
```



17.4 RecyclerView

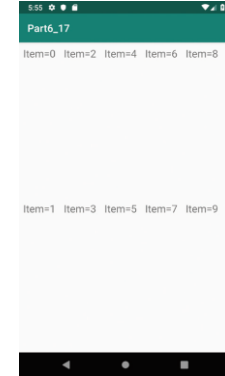
- GridLayoutManager

```
GridLayoutManager gridManager=new GridLayoutManager(this, 2);  
recyclerView.setLayoutManager(gridManager);
```



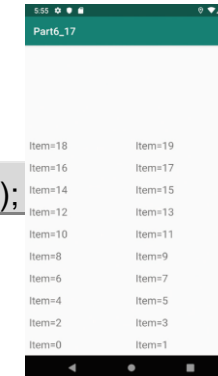
- 방향 지정

```
GridLayoutManager gridManager=new GridLayoutManager(this, 2, GridLayoutManager.HORIZONTAL, false);
```



- 아래부터 나열

```
GridLayoutManager gridManager=new GridLayoutManager(this, 2, GridLayoutManager.VERTICAL, true);
```



- StaggeredGridLayoutManager

```
StaggeredGridLayoutManager sgManager=new StaggeredGridLayoutManager(2,  
StaggeredGridLayoutManager.VERTICAL);
```



17.4 RecyclerView

17.4.4. ItemDecoration

각 항목을 다양하게 꾸미기

- onDraw: 항목을 배치하기 전에 호출
- onDrawOver: 모든 항목이 배치된 후에 호출
- getItemOffsets: 각 항목을 배치할 때 호출

```
class MyItemDecoration extends RecyclerView.ItemDecoration {  
  
    @Override  
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,  
                               RecyclerView.State state) {  
        super.getItemOffsets(outRect, view, parent, state);  
        //항목의 index 값 획득  
        int index=parent.getChildAdapterPosition(view)+1;  
  
        if(index % 3 == 0)  
            //left, top, right, bottom  
            outRect.set(20, 20, 20, 60 );  
        else  
            outRect.set(20, 20, 20, 20 );  
  
        view.setBackgroundColor(0xFFECE9E9);  
        ViewCompat.setElevation(view, 20.0f);  
    }  
}
```

17.4 RecyclerView

- ItemDecoration을 RecyclerView에 적용

```
recyclerView.addItemDecoration(new MyItemDecoration());
```

- onDraw() 함수

```
public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {  
    super.onDraw(c, parent, state);  
    //RecyclerView의 사이즈 계산  
    int width=parent.getWidth();  
    int height=parent.getHeight();  
  
    Paint paint=new Paint();  
    paint.setColor(Color.RED);  
    c.drawRect(0, 0, width/3, height, paint );  
    paint.setColor(Color.BLUE);  
    c.drawRect(width/3, 0, width/3*2, height, paint);  
    paint.setColor(Color.GREEN);  
    c.drawRect(width/3*2, 0, width, height, paint);  
}
```

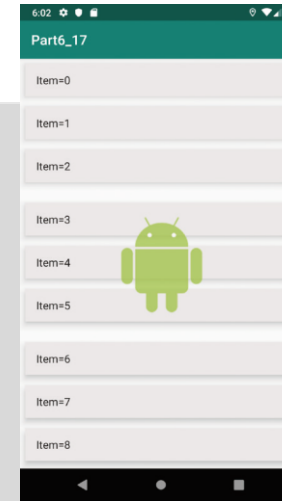


17.4 RecyclerView

- onDrawOver()

```
public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
    super.onDrawOver(c, parent, state);
    //RecyclerView의 사이즈 계산
    int width=parent.getWidth();
    int height=parent.getHeight();
    //이미지 사이즈 계산
    Drawable dr= ResourcesCompat.getDrawable(getResources(),
                                                R.drawable.android, null);
    int drWidth=dr.getIntrinsicWidth();
    int drHeight=dr.getIntrinsicHeight();

    int left=width/2 - drWidth/2;
    int top=height/2 - drHeight/2;
    c.drawBitmap(BitmapFactory.decodeResource(getResources(),
                                                R.drawable.android), left,top,null);
}
```



Step by Step 17-4 - RecyclerView

- 액티비티 생성
- 파일 복사
- Support 라이브러리 dependency 추가
- activity_lab17_4.xml 작성
- Lab17_4Activity 작성
- 실행

