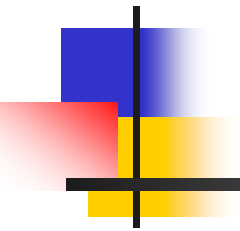


Fragment

배 희호 교수
경복대학교
소프트웨어융합과

Activity

- Android Application을 구성하는 주요 Component의 하나이며, 각 예제마다 하나씩 Activity를 만들었는데 각 예제의 화면 하나가 바로 Activity 임
- Window와 유사한 개념이지만 “하나의 화면”이라고 이해하는 것이 옳음
  - 즉, Activity는 사용자와 상호 작용할 수 있는 하나의 Window라고 생각하면 옳음
- 사용자와의 Interface를 구성하지만 그 자체는 출력 기능이 없으므로 직접적으로 보이지는 않음

Activity

- Activity와 View
  - 사용자 눈에 실제로 보이는 것은 View이며, Activity는 반드시 내부에 View나 ViewGroup을 가져야 함
  - Activity가 생성될 때마다 호출되는 setContentView() 메소드가 Activity안에 View를 배치하는 명령 임
  - 실제 Application에서는 한 화면에서 복잡한 동작을 다 수행할 수 없으므로 기능별로 작업을 실행할 수 있는 여러 개의 Activity가 필요함
  - 또한 여러 개의 Activity를 생성하였기 때문에 Activity간에 통신할 수 있는 방법이 필요함

- 다수의 Activity를 만드는 방법
  - 화면을 전환하는 경우(Intent)
  - 화면을 분할하는 경우(Fragment)

Activity

특징	Activity	View
역할	하나의 화면을 나타내는 Component	개별적인 UI 요소
책임	생명 주기 관리, 사용자 인터페이스 관리	화면 그리기, 사용자 입력 처리
구성 요소	여러 View를 포함하는 컨테이너 역할	Button, Text, Image 등 개별 UI 요소
생명 주기 관리	onCreate(), onStart(), onResume() 등	N/A
사용 예시	로그인 화면, 메인 화면 등	Button, TextView, ImageView등
정의 위치	Java/Kotlin Code	XML Layout File 또는 Java/Kotlin Code

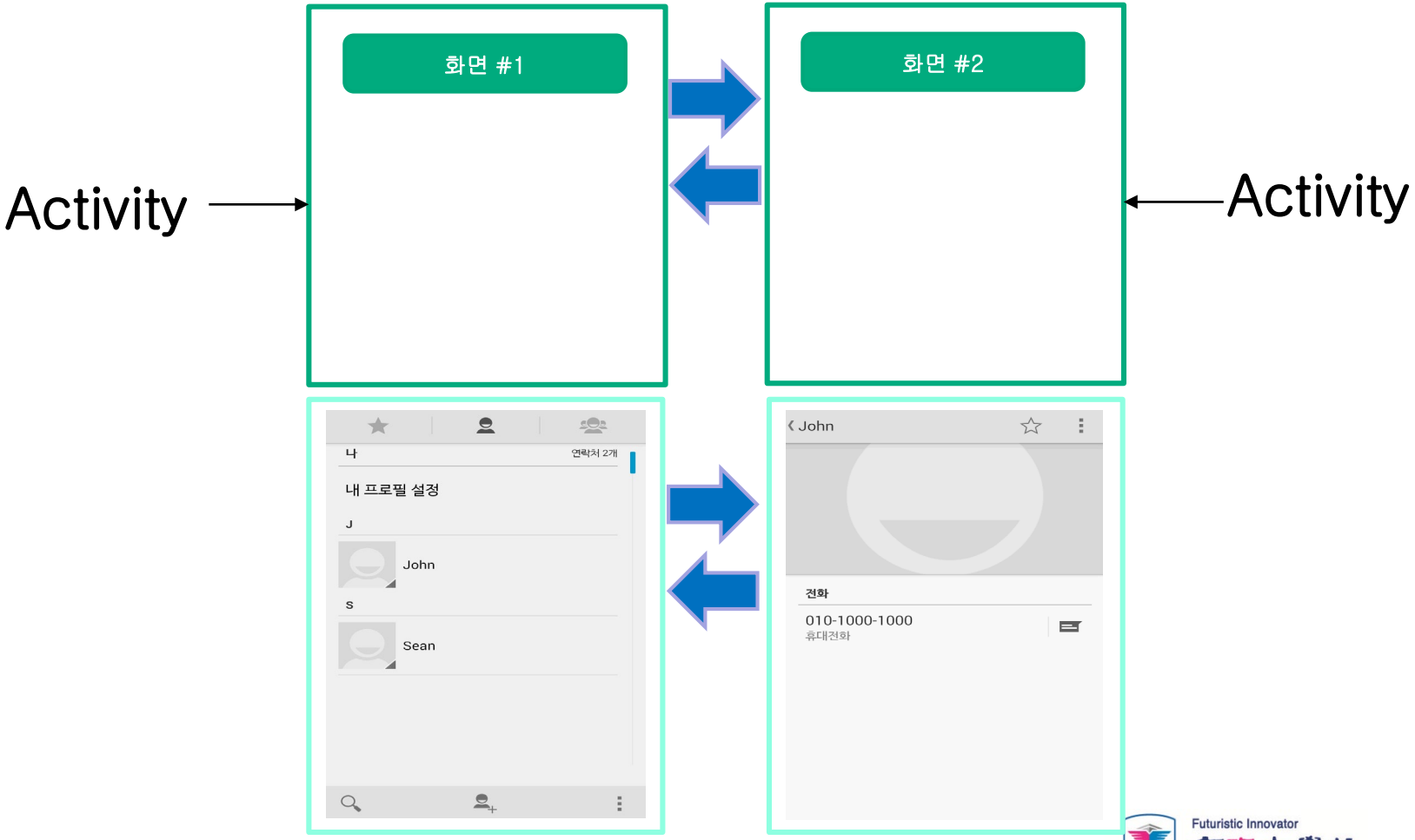
Activity

- Android App에서 다양한 화면 전환 및 UI 구성 요소

요소	설명	주요 사용 사례
Intent	Component 간 Message 전달을 위한 객체	✓ Activity 전환 ✓ Activity 간의 Data 전달
Fragment	Activity 내 화면의 일부 구성	✓ 화면 일부만 변경 ✓ UI 모듈화
Tab	화면 전환을 위한 UI 구성 요소	✓ 동일 주제의 화면 간 빠른 전환

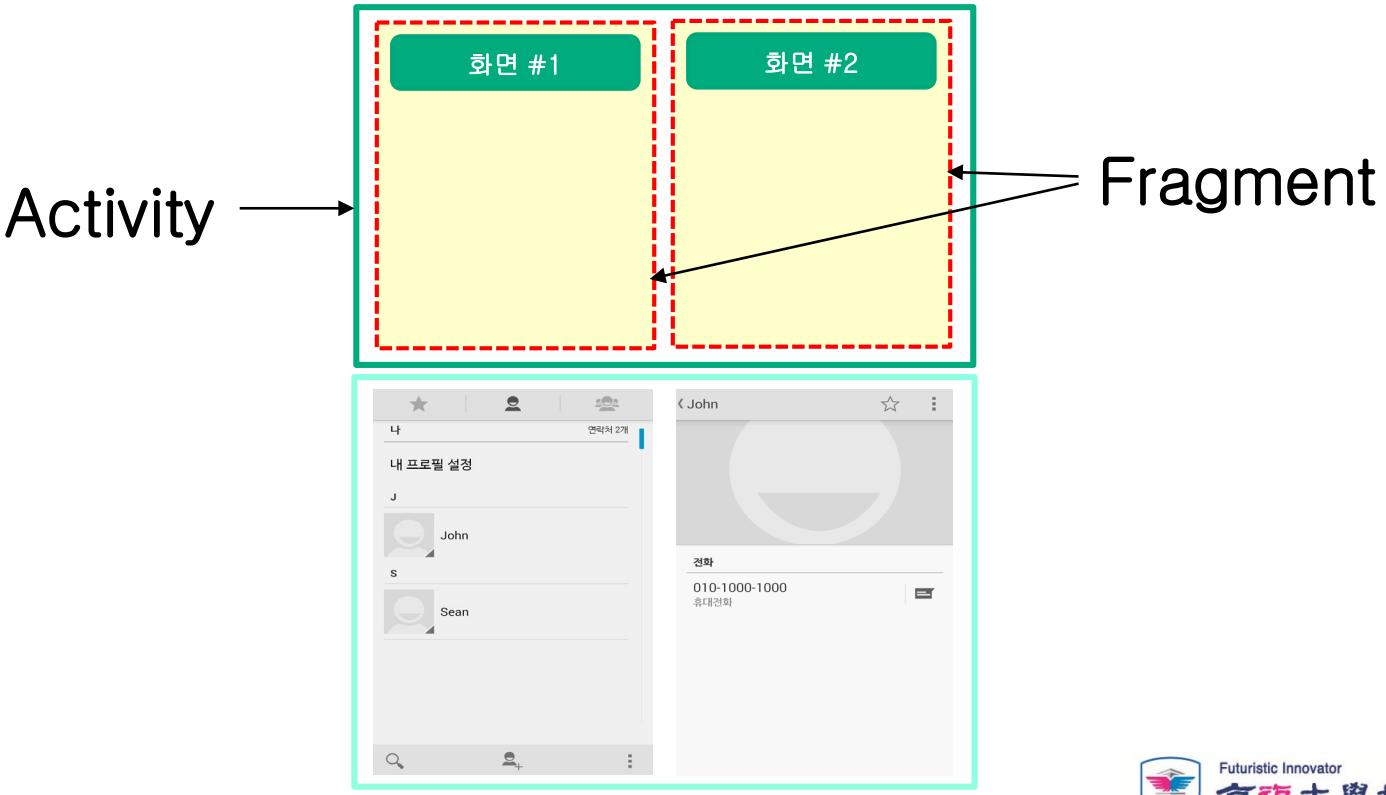
Activity

- 화면을 전환하는 경우
  - 2개의 화면을 Activity로 만들고 Activity간 전환



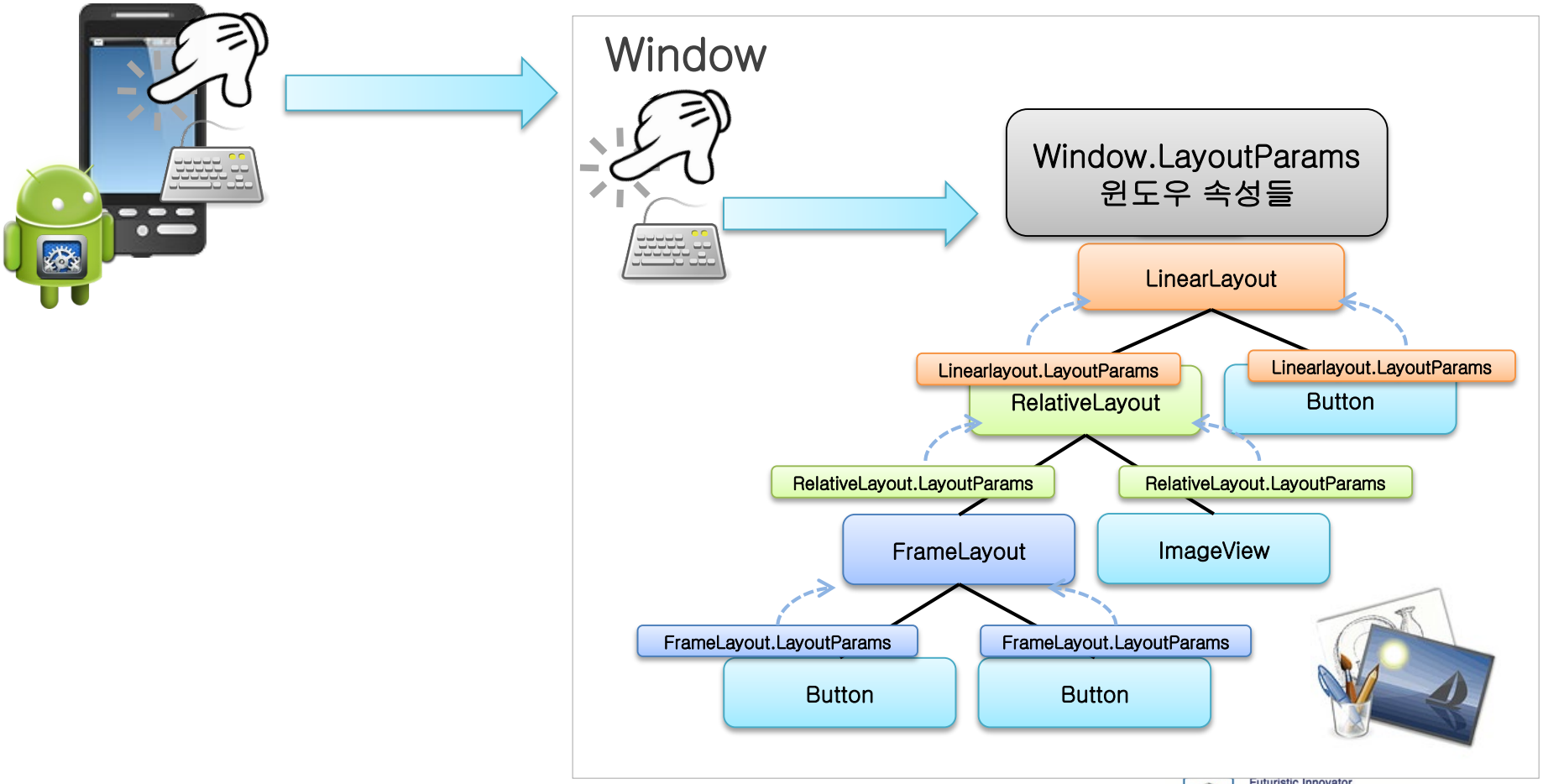
Activity

- 하나의 Activity위에 Fragment를 두고 Fragment간 전환
  - Fragment로 만들어 두면 SmartPhone에서는 Fragment 간 화면 전환, Tablet에서는 2개의 Fragment를 하나의 Activity위에서 동시에 보여주는 화면 분할이 가능함



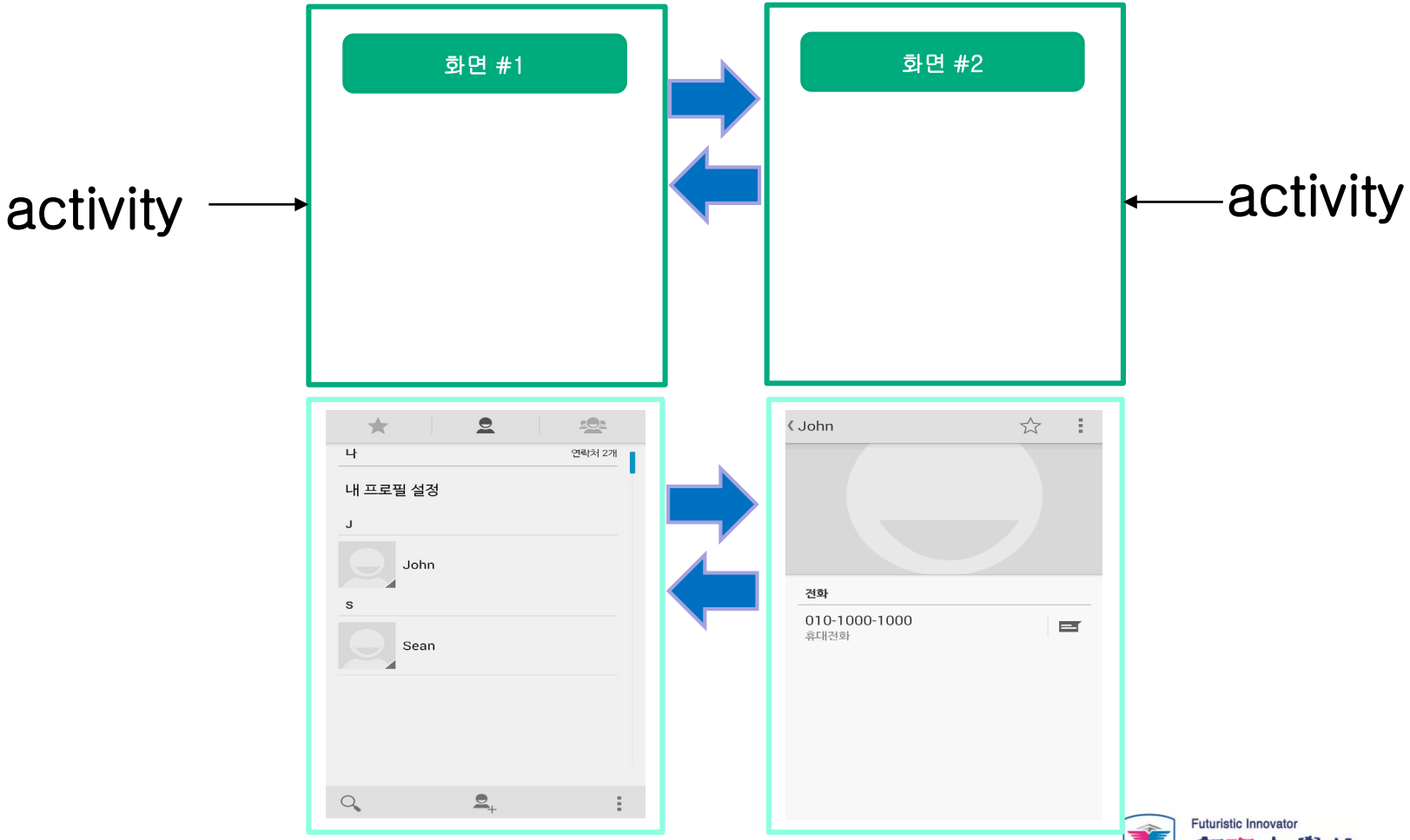
Window와 View

- Window는 뭔가를 그릴 수 있는 창이며, 보통 하나의 Surface를 가지고 있음



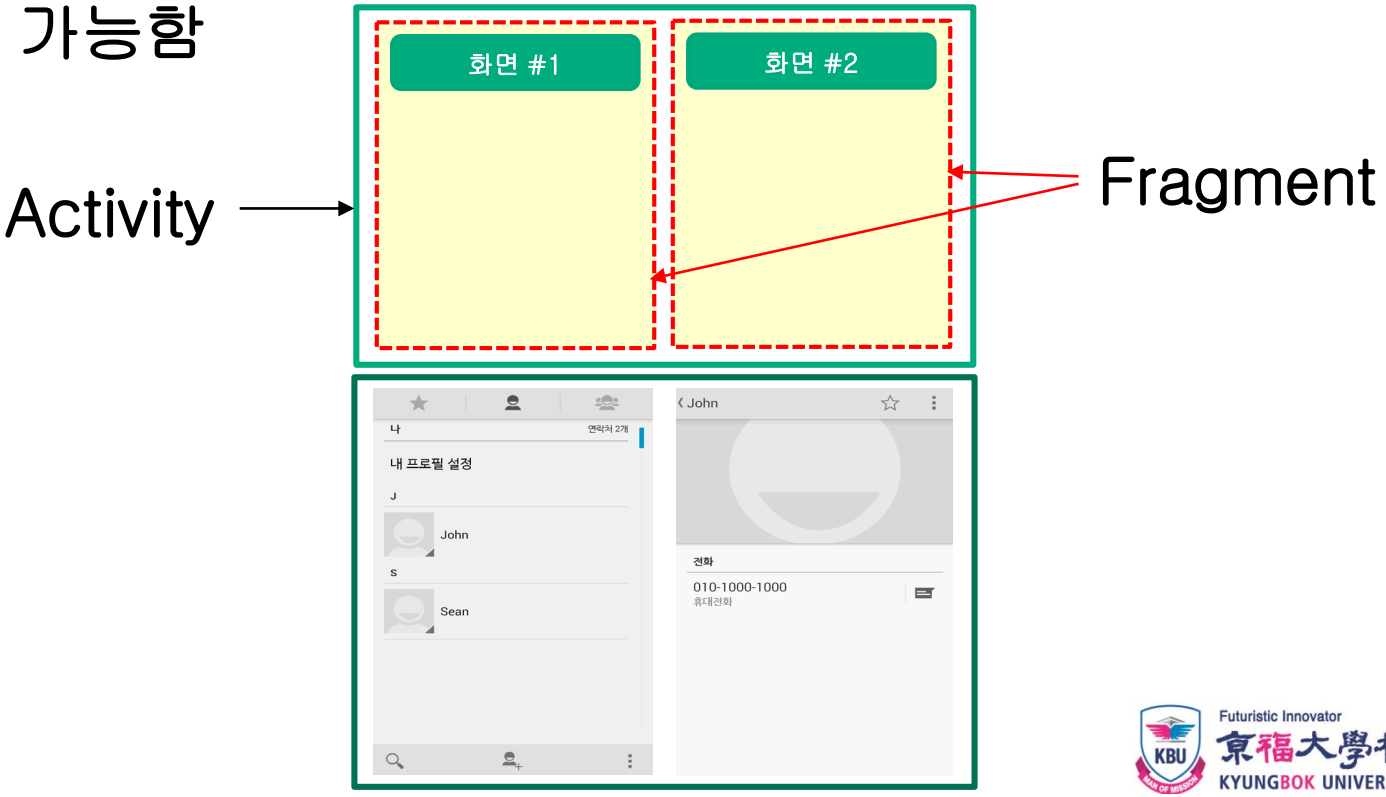
Window와 View

- 화면을 전환하는 경우(Intent)
  - 두 개의 화면을 Activity로 만들고 Activity 간 전환

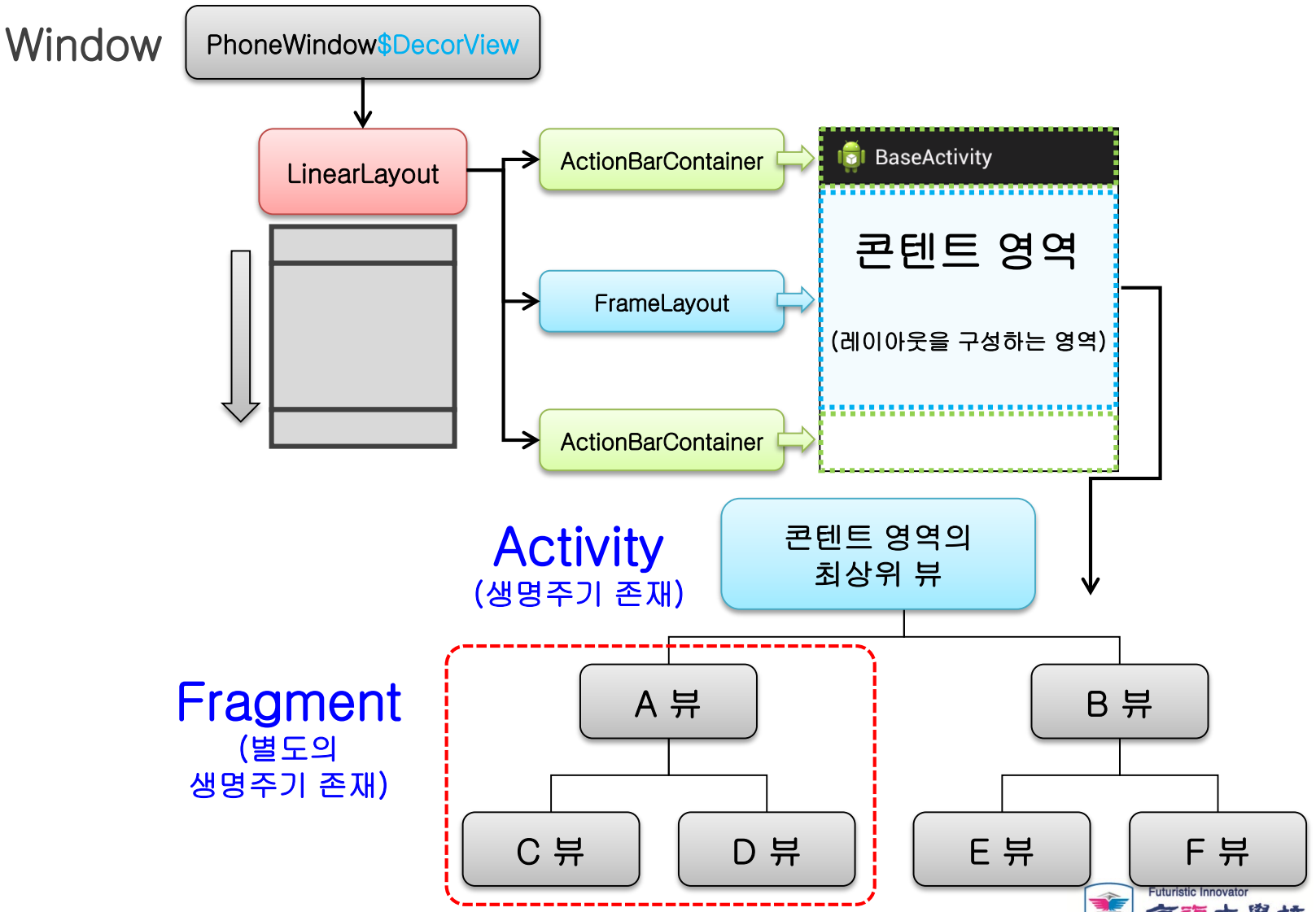


Window와 View

- 화면을 분할하는 경우(Fragment)
  - 하나의 Activity위에 Fragment를 두고 Fragment 간 전환
    - Fragment로 만들어 두면 Smart Phone에서는 Fragment간 화면 전환, Tablet에서는 2개의 Fragment를 하나의 Activity위에서 동시에 보여주는 화면 분할이 가능함



Activity와 Fragment



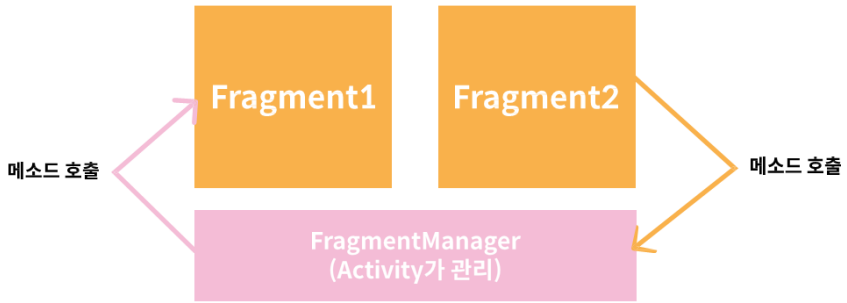
Activity와 Fragment

Activity

AndroidMenifest파일에 Activity 컴포넌트 등록

독립적으로 존재할 수 있음!
(안드로이드 시스템에서 관리해서)

Fragment

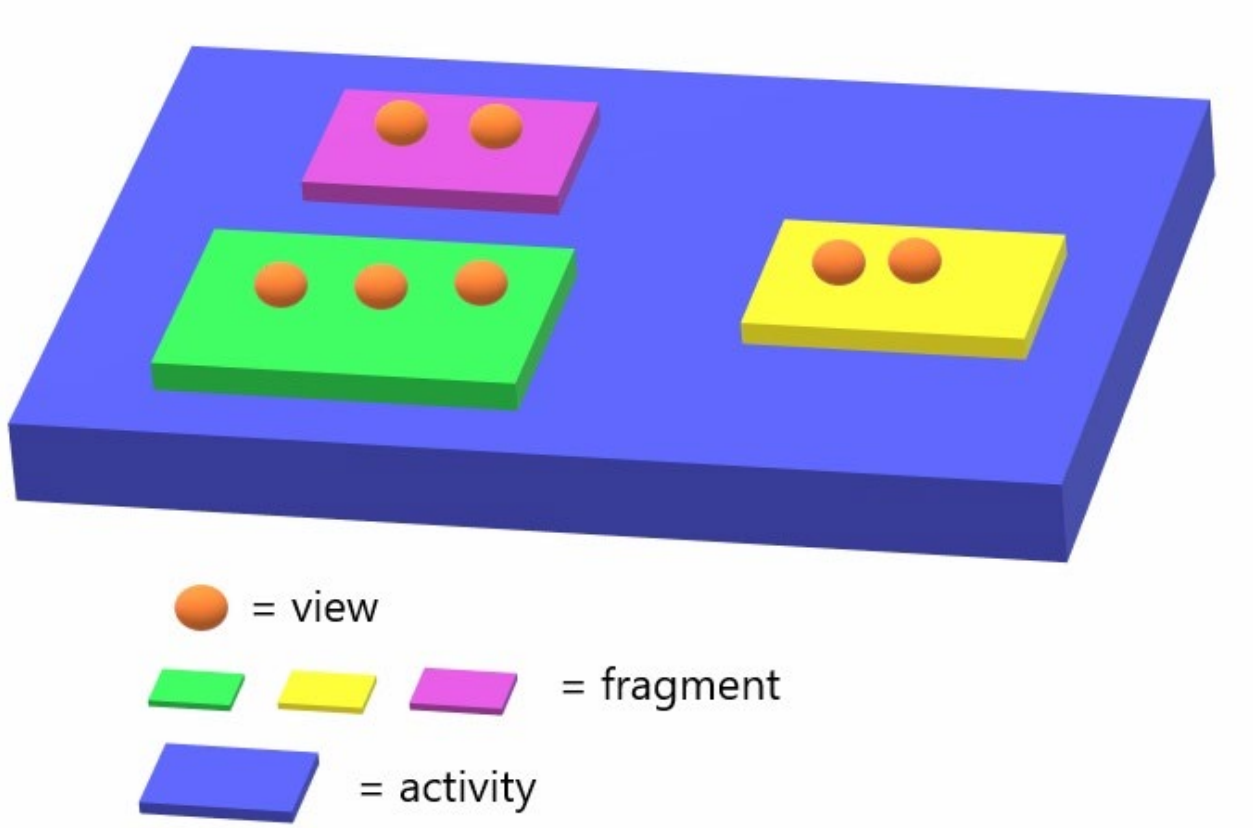


AndroidMenifest파일에 Fragment 컴포넌트 등록 X

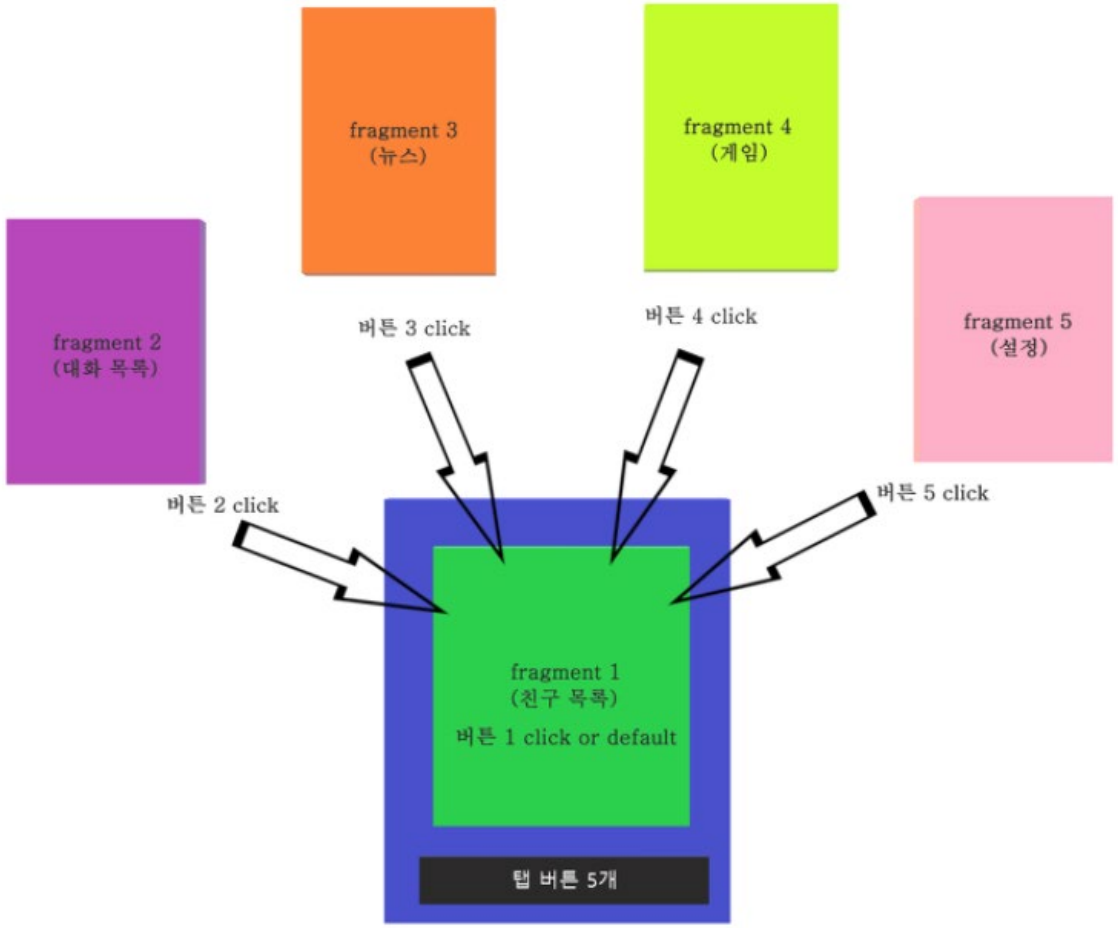
독립적으로 존재할 수 없음!
(Activity나 상위 Fragment에 종속적이기 때문에)

Activity와 Fragment

- 기존에는 한 개의 Activity위에 여러 개의 View를 지정하여 Activity를 만들었다면, Activity 위에 Fragment를 올리고 Fragment 위에 View를 올리는 것

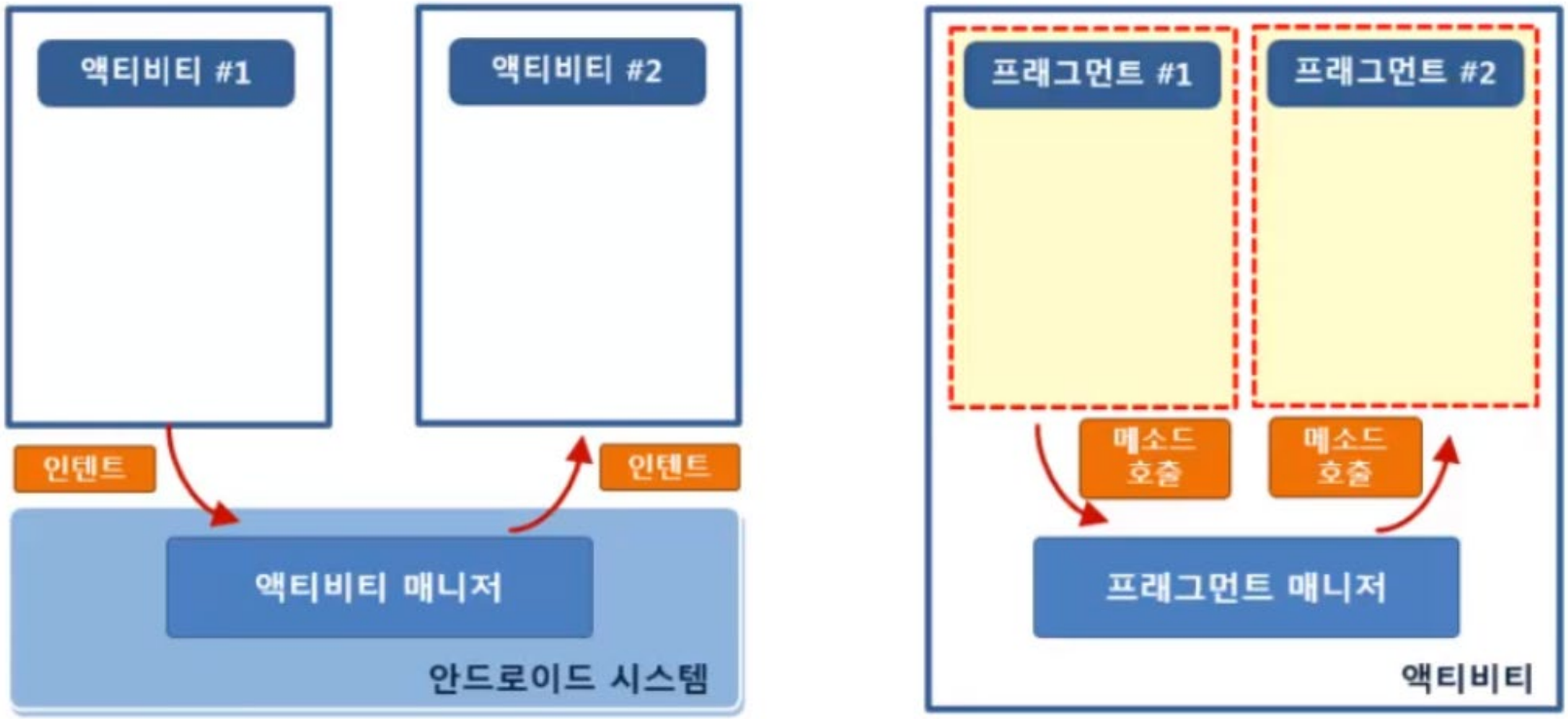


Activity와 Fragment

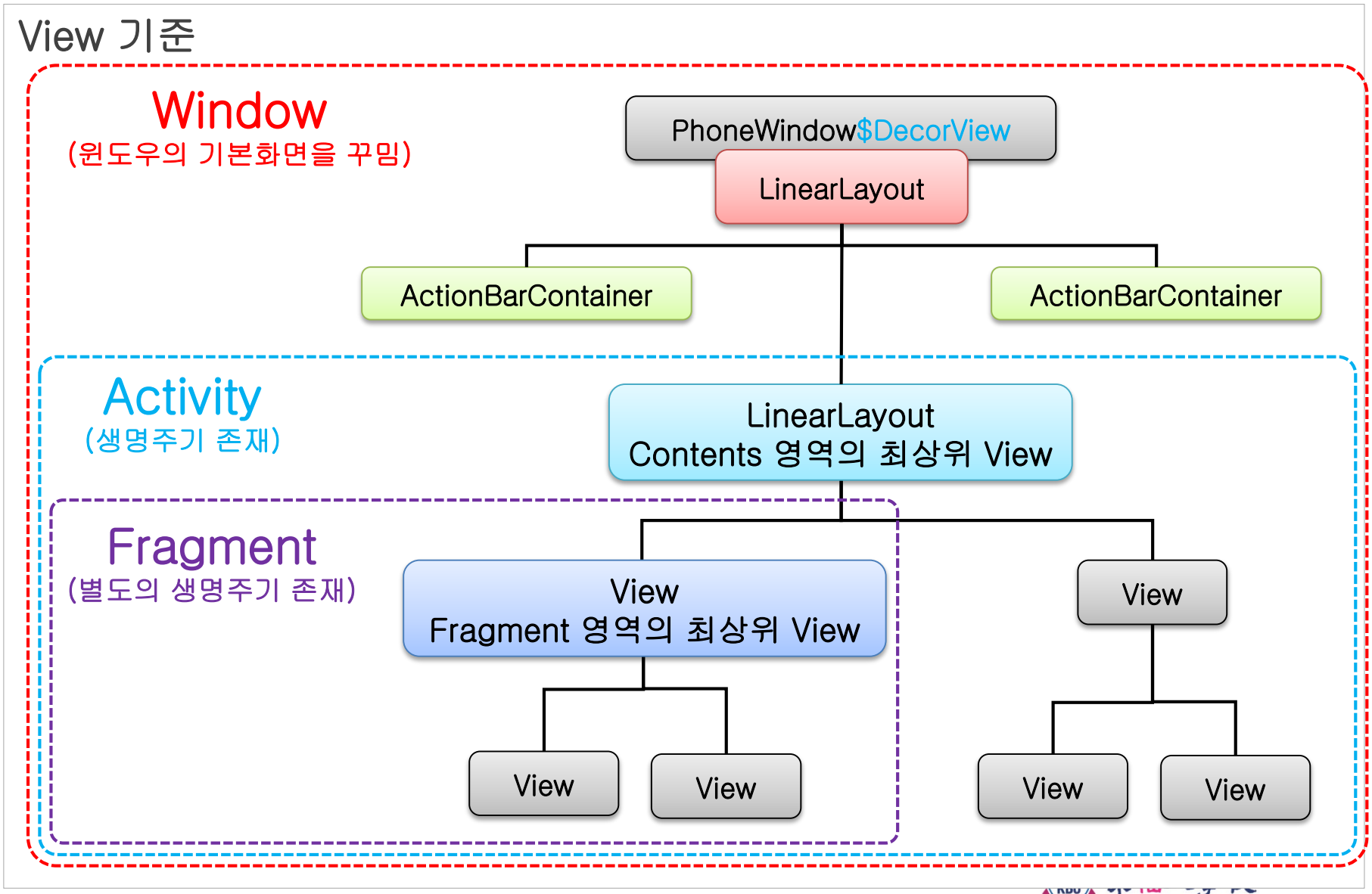


Activity와 Fragment

- Activity를 System의 Activity Manager에서 관리하듯이 Fragment를 Activity의 Fragment Manager에서 관리함
- Fragment 입장에서는 Activity가 System 역할을 하므로 Fragment는 항상 Activity 위에 올라가 있어야 함



Activity와 Fragment

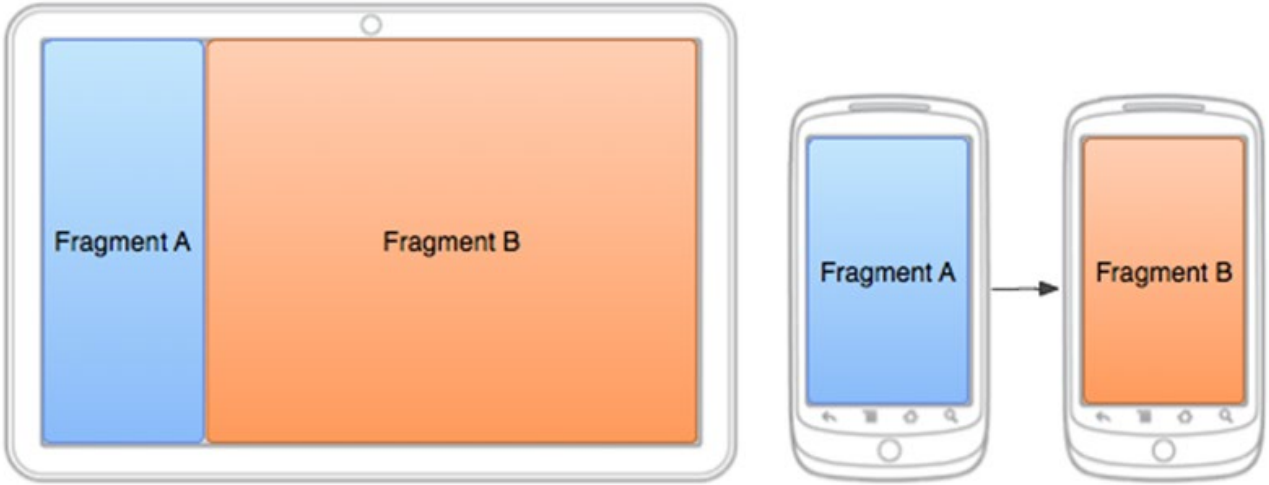


Activity와 Fragment

- 결국 화면은 오로지 View로만 구성
  - Window는 전체 View를 가진 객체이며, Window로 전달되는 각종 사용자 Event를 전체 View에 전달
  - Activity는 Window가 가진 전체 View의 일부분(Contents 영역) 하위에 자식 View들을 추가할 수 있고, 별도의 생명 주기를 가짐
  - Fragment는 Activity가 가진 전체 View의 일부분 하위에 자식 View들을 추가할 수 있고, Activity 생명 주기 내에 자신만의 또 다른 생명 주기를 가짐. 따라서 마치 Activity의 자식 Activity처럼 사용됨

Fragment

- Fragment의 사전적 의미 또한 '조각', '파편'임
- Fragment는 하나의 Activity 내에서 UI의 일부를 나타내는 Module화된 구성 요소
- Activity보다 작은 단위의 화면
- Android가 Fragment를 처음 도입한 것은 Android 3.0(API 레벨 11)부터 임

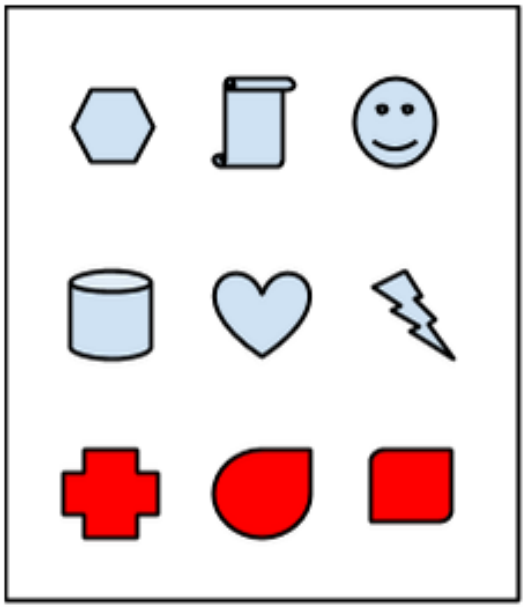


(*그림 출처: developer.android.com)

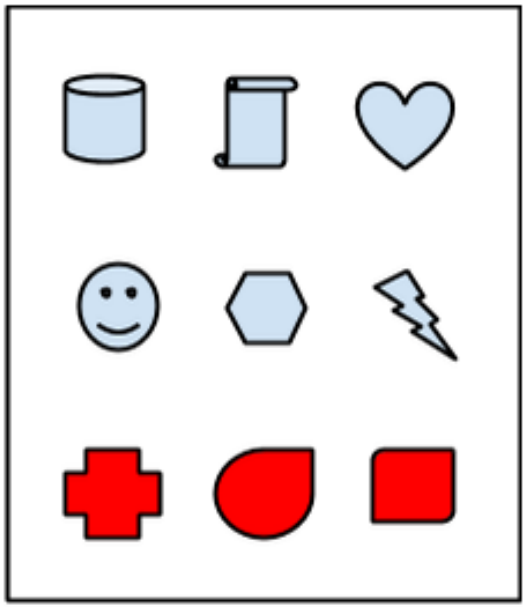
https://developer.android.com/guide/components/fragments.html?hl=ko

Fragment

- 등장 배경
  - Activity A와 Activity B는 서로 다른 UI를 가지지만 일부 같은 UI를(빨간색) 가짐
  - 이런 경우 같은 UI 부분을 공통 Module 빼고 싶음
    ⇒ 공통 UI 부분을 Fragment



Activty A



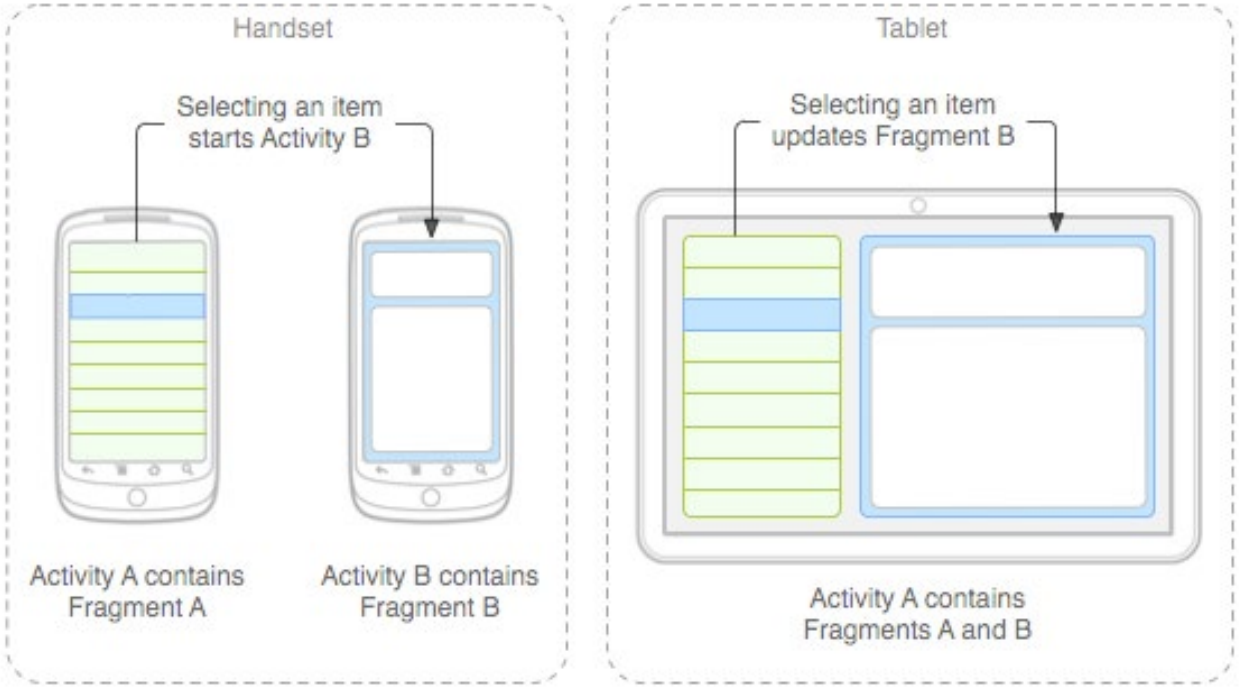
Activty B

Fragment

- Fragment의 등장은 Tablet의 등장과 관련되어 있음
  - Activity X와 Y는 일반적인 Smart Phone UI를 구성하고 있음
  - Activity Q는 Landscape(가로) 화면일 때, 보여주고 싶은 화면
  - 가끔 Tablet같은 대화면 Device가 나타나면 Portrait(세로) 화면이라도 Activity P와 같은 화면을 보여주고 싶음
  - Activity X / Activity Y의 UI를 Fragment화 함
  - Q와 P에서는 Fragment를 이용하는 형태로 구현
- Fragment를 사용하면 Code를 재사용하여 동시에 여러 크기의 기기 화면 크기를 지원해 줄 수 있음

Fragment

- Fragment를 사용하면 대형화면에서 Activity 화면을 분할해서 표현 가능
- Smart Phone과 같은 소형 화면에서는 화면의 분할보다는 실행 중에 화면을 동적(Dynamic)으로 추가, 제거하는 데 더 많이 활용



Fragment

- Activity의 기본 개념은 한 화면에 보이는 모든 것을 관리 하는 개념임
- Fragment는 Activity 실행 중에 추가 및 제거가 가능한 Module식 Section
- Fragment는 자체 Lifecycle를 가지고, 재사용 가능하고 다른 액티비티에서도 사용할 수 있음
- 자체 입력 Event를 받음
- 다양한 Tablet Device, Dynamic한 Application 개발을 위해서 Activity만으로는 개념에 맞지 않아 등장 한 것이 Fragment라는 개념
  - 기존 Activity는 하나의 화면에 여러 개 사용 할 수 없게 설계되어 있는 반면, Fragment는 Activity와 비슷한 Lifecycle을 가지면서 여러가지 화면을 넣을 수 있는 방법을 지원함

Fragment

- Fragment의 기본 목적
  - 하나의 화면이 XML Layout과 JAVA Source로 구성된다는 점에 착안하여 하나의 Fragment가 XML Layout과 JAVA Source로 구성되도록 하고 독립적으로 관리되도록 하기 위함
- 주의할 점
  - Fragment를 사용하지 않고 일반 XML Layout으로 구성하는 경우에 비해 Code의 양이 많아지고 복잡해질 수 있음

Fragment

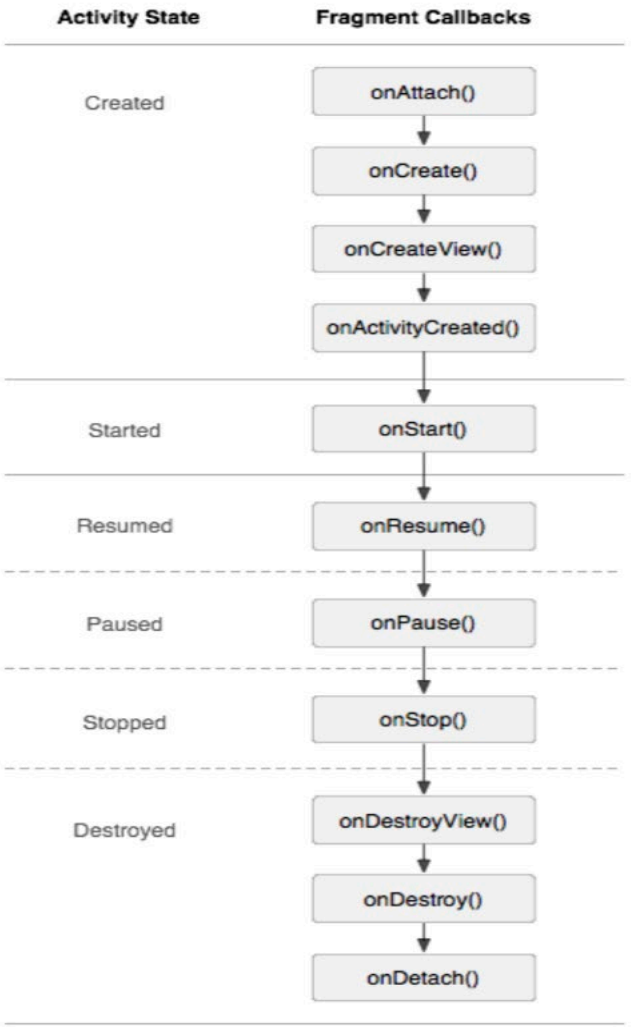
- 특징
  - Tablet의 경우 Activity를 사용하면 많은 공간이 낭비 됨
  - 이런 공간 낭비를 막기 위해 View를 구성하게 되지만, 이런 관리들이 너무 복잡하게 되며 사용자들의 혼란을 초래할 수 있음
  - SmartPhone의 경우 ViewPager를 이용하여 작은 화면을 사용자들이 좀 더 Dynamic하게 사용 할 수 있는 용도로 바꿀 수 있음
  - Fragment는 Tablet과 좀더 Dynamic한 Application 개발에 필수로 사용되므로 Android 개발자라면 반드시 알아야 함

Fragment

- 특징
  - Fragment는 Activity와 비슷한 LifeCycle을 가짐
  - Fragment는 하나의 Activity에서 다수의 Fragment를 사용 할 수 있음
  - Fragment는 Activity에서만 존재하며 단독으로 실행 될 수 없는 구조
  - Fragment는 Activity와 마찬가지로 Back Stack을 사용 할 수 있으나, Activity처럼 다양한 Stack 방식을 지원하지 않음
  - Fragment는 Activity위에서만 존재 하기 때문에 다수의 Fragment를 동시에 띄울 때 Memory가 문제가 될 수 있으므로 너무 복잡한 구조는 지양해야 함

Fragment

- Lifecycle
  - Activity가 Fragment를 감싸고 있는 구조
  - 기본적인 Lifecycle은 Activity의 Lifecycle과 관계가 있을 수 밖에 없음
  - Fragment의 생명주기는 Activity 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);



Fragment

- Lifecycle
  - onAttach(Activity activity)
    - Fragment가 Activity에 포함되는 순간 호출
  - onCreate(Bundle savedInstanceState)
    - Activity의 onCreate() 함수와 동일
  - onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
    - Fragment의 UI 구성을 위해 호출
    - 이곳에서 반환하는 View가 Fragment 화면에 출력
  - onActivityCreated(Bundle savedInstanceState)
    - Fragment의 Activity 생성이 완료된 순간 호출
  - onStart()
    - Activity의 onStart() 함수와 동일
  - onPause()
    - Activity의 onPause() 함수와 동일

Fragment

- Lifecycle
  - onResume()
    - Activity의 onResume() 함수와 동일
    - 사용자 화면에 출력되면서 사용자와 상호 작용이 가능한 상태
  - onStop()
    - Activity의 onStop() 함수와 동일
  - onDestroyView()
    - Fragment가 화면에서 사라진 후 BackStack에 추가된 후 호출
  - onDestroy()
    - Activity의 onDestory() 함수와 동일
  - onDetach()
    - Fragment가 Activity에서 제거될 때 호출

Fragment

- Fragment는 Activity와 달리 onAttach()와 onDetach() 메소드가 있음
  - Fragment는 Activity 위에 올라갔을 때 Fragment로 동작할 수 있기 때문에, Activity 위에 올라갈 때 onAttach(), Activity에서 내려올 때 onDetach() 메소드를 자동으로 호출해줌으로써 그 시점을 알 수 있도록 해줌

Fragment

- Activity Created
  - onAttach() 메소드
    - Fragment가 Activity에 최초로 연결될 때 호출
    - Fragment를 붙이는 방법에 따라서 호출되는 시점이 다른데 Static하게 붙였다면 Activity가 시작될 때 같이 호출이 되고 Dynamic하게 붙였다면 FragmentManager에 의해서 Activity에 연결될 때 호출
  - onCreate() 메소드
    - Fragment를 초기화하는 메소드
    - 최초로 생성될 때 호출

Fragment

- Activity Created
  - onCreateView() 메소드
    - 여기서 Layout inflate 작업이 진행

```
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container, Bundle savedInstanceState) {
   // Inflate the layout for this fragment
   return inflater.inflate(R.layout.example_fragment, container, false);
}
```

  - onActivityCreated() 메소드
    - Activity의 onCreate()에서 UI 작업이 마무리 된 이후 시점이라고 생각하면 됨
    - Fragment가 Activity에 완벽하게 연결이 된 상태

Fragment

- Activity Started
  - onStart() 메소드
    - 부모 Activity가 화면에 보이게 되면 호출
- Activity Resumed
  - onResume()
    - 부모 Activity가 유저 Input을 받을 준비가 되면 호출
- Activity Paused
  - onPause()
    - 부모 Activity가 화면에는 보이지만 Focus를 잃게 되면 호출
- Activity Stopped
  - onStop()
    - 부모 Activity가 더이상 화면에 보이지 않게 되면 호출

Fragment

- Activity Destroyed
  - onDestroyView()
    - onCreateView()에서 호출된 View들이 Activity에서 제거되면서 호출
    - 일반적으로 View Resource를 해제하는데 사용
  - onDestroy()
    - onCreate()에 대응되는 함수로 Fragment가 더이상 유효하지 않을 때 호출
    - 일반적으로 Fragment 자체 Resource를 해제하는 용도로 사용
  - onDetach()
    - Fragment가 더이상 Activity에 연결되어 있지 않은 상황에서 호출
    - 일반적으로 부모 Activity에서 Fragment의 참조를 가지고 있다면 null로 바꿔주는 작업을 수행

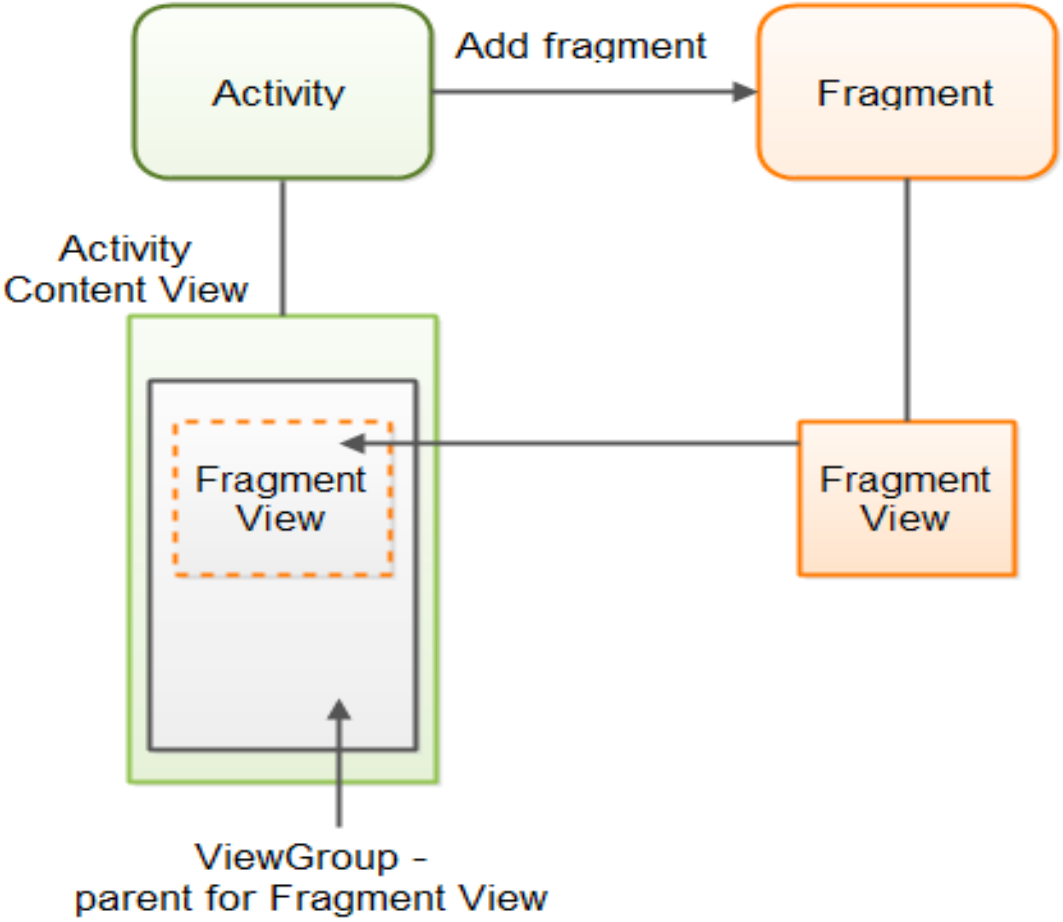
Fragment

- Fragment는 자신이 속한 Activity와만 통신을 해야만 하며, 항상 자신이 속한 Activity를 통해서 다른 Fragment나 Activity와 통신해야 함
- Fragment와 Activity가 통신할 수 있는 3가지 방법
  - Bundle
    - Activity는 Fragment를 생성 후, Data를 넣은 bundle을 전달할 수 있음
    - Fragment는 onActivityCreated() 메소드에서 bundle을 받게 됨
  - Method
    - Activity는 Fragment의 메소드를 호출 할 수 있음
  - Listener
    - Fragment는 Interface를 사용하여 Activity에서 Listener Event를 발생시킬 수 있음

Fragment

- Activity에 붙이는 방법
  - Static 방법
    - Layout File에 선언해서 붙이기
  - Dynamic 방법
    - FragmentManager를 통해서 붙이기
  - 공통적으로 Fragment 클래스를 상속받아 Fragment를 생성

Fragment



Fragment

- Static 방법
  - Activity XML Layout File에 <fragment> tag 추가하여 Fragment를 정의
  - Fragment의 Layout은 Activity에서 inflate되어 ViewGroup이 됨
  - onInflate() 메소드 호출 시 Fragment Lifecycle이 시작
  - 처음 화면에 출력될 Fragment 지정이나 화면에 동적 변화 없을 경우 사용
  - android:name 속성에 Fragment 클래스를 명기해야 함
  - 실행 중에 Fragment를 추가/삭제/변경할 수 없음

Fragment

- Static 방법
  - 기존의 View를 붙이는 것과 다를 게 없음
  - Activity가 사용하는 Layout에 Fragment를 그대로 붙이는 방법

```
<?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="match_parent"
   android:orientation="vertical" >
   <fragment
      android:id="@+id/fragment"
      android:name="example.fragment"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>
</LinearLayout>
```

Fragment

- Dynamic 방법
  - Activity 실행 중에 Fragment를 추가/삭제/변경
  - Activity XML Layout 파일에 <FrameLayout> tag 추가 후, Programming적으로 FragmentManager를 이용하여 ViewGroup에 Fragment를 동적으로 추가
  - onAttach() 콜백 메소드에서 Fragment Lifecycle이 시작

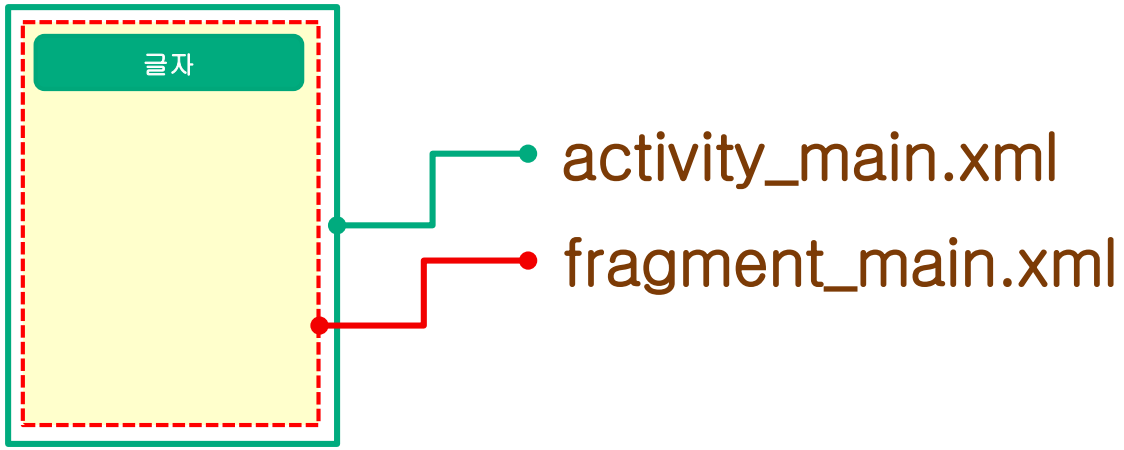
Fragment

- Dynamic 방법
  - Activity가 이미 화면에 보이는 상황에서 Runtime으로 Fragment를 붙이는 방법
  - 절차
    - FragmentManager의 객체 생성
    - FragmentTransaction을 시작 (beginTransaction())
    - Fragment를 붙임
    - FragmentTransaction을 commit() 함

```
FragmentManager fragmentManager = getFragmentManager();
TestFragment frament = new TestFragment();
Bundle bundle = new Bundle();
frament.setArguments(bundle);
FragmentTransaction transaction = fragmentManager.beginTransaction();
transaction.add(R.id.container, fragment);
transaction.commit();
```

Fragment

- Single Fragment
  - 하나의 Activity에 하나의 Fragment가 들어가 있는 형태
  - 하나의 Activity안에서 Fragment를 교체하거나 추가 하기 좋음
  - PlaceholderFragment라는 이름으로 만들어짐



Fragment

- 사용 방법
  - Fragment 클래스
    - Fragment의 실체는 Activity와 크게 다를 것이 없음
    - [JAVA 클래스 + XML Layout]이 그 실체
    - 만약 Static하게 Activity에 붙인다면, XML Layout만 있어도 됨. 하지만 Code Level의 분리를 위해서는 [JAVA 클래스 + XML Layout]를 선택
  - FragmentManager
    - Fragment를 관리하는 객체
  - FragmentTransaction
    - Fragment의 처리를 위해 만든 단위

Fragment

- Fragment의 종류
  - ListFragment
    - ListFragment는 ListView로 화면을 구성할 때 ListView와 관련된 내용만 Activity에서 분리해서 Fragment로 구현할 수 있게 만든 클래스
    - ListView에서 다룰 수 있는 onListItemClick()과 같은 콜백 함수들도 제공
  - WebViewFragment
    - WebViewFragment는 WebView를 포함하는 Fragment
    - WebViewFragment에 포함된 WebView에서 원하는 Web Page를 표시할 수 있음

Fragment

- Fragment의 종류
  - DialogFragment
    - DialogFragment는 Dialog를 표시하는 Fragment
    - DialogFragment는 Fragment 클래스를 확장하는 Utlity Class
    - Fragment는 Back Stack에 넣어둘 수 있기 때문에 사용자가 다시 Fragment로 복귀하고자 할 때에 Activity에 기본적으로 들어있는 Dialog 대신에 사용할 수 있는 좋은 대체제임
  - PreferenceFragment
    - Preference 객체들을 목록으로 보여주는 PreferenceActivity와 비슷하며, App의 Settings를 만들 때에 유용하게 사용할 수 있음

Fragment

- ListFragment

```
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);
  }
}
```

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

Fragment

- WebViewFragment

```
public class TwoFragment extends WebViewFragment {
    @Override
    public void onViewCreated(View view, Bundle savedInstanceState) {
            super.onViewCreated(view, savedInstanceState);
            WebView webView=getWebView();
            WebSettings settings=webView.getSettings();
            settings.setJavaScriptEnabled(true);
            webView.setWebViewClient(new WebViewClient());
            webView.loadUrl("http://www.google.com");
    }
}
```

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);
```

Fragment

- Fragment는 자신이 속한 Activity와만 통신을 해야만 하며, 항상 자신이 속한 Activity를 통해서 다른 Fragment나 Activity와 통신해야 함
- Fragment와 Activity가 통신할 수 있는 3가지 방법
  - Bundle
    - Activity는 Fragment를 생성 후, Data를 넣은 Bundle을 전달할 수 있음
    - Fragment는 onActivityCreated() 메소드에서 Bundle을 받게 됨
  - Method
    - Activity는 Fragment의 메소드를 호출 할 수 있음
  - Listener
    - Fragment는 Interface를 사용하여 Activity에서 Listener Event를 발생시킬 수 있음

Fragment

- FragmentTransaction를 통해 Fragment를 붙일 때에는 아래 선택 사항이 있음
  - add() 메소드 이용
    - 기존에 Fragment가 존재한다면 계속 쌓임
      ⇒ 리소스 낭비/성능저하
  - replace() 메소드 이용
    - 기존의 Fragment를 제거하고 붙음

Fragment

- Fragment 특징
  - 기존 Activity는 하나의 화면에 여러 개 사용할 수 없게 설계되어 있는 반면 Fragment는 여러 가지 화면을 넣을 수 있는 방법을 지원
    - Fragment는 Activity와 비슷한 LifeCycle을 가짐
    - Fragment는 하나의 Activity에서 다수의 Fragment를 사용 가능
    - Fragment는 Activity에서만 존재하며 단독으로 실행될 수 없는 구조
    - Fragment는 Activity와 마찬가지로 Back Stack을 사용할 수 있으나, Activity처럼 다양한 Stack 방식을 지원하지 않음
    - Fragment는 Activity와 위에서만 존재하기 때문에 다수의 Fragment를 동시에 띄울 때 Memory 문제가 될 수 있으므로 너무 복잡한 구조는 지양해야 함

Fragment 클래스

- Fragment는 Activity에 포함되는 Sub Activity라고 볼 수 있음
- 하나의 Activity에 여러 개의 Fragment를 배치할 수 있음
- Fragment는 자체 Lifecycle을 가지고 있음
- Activity가 실행중인 동안 Fragment를 추가, 삭제할 수 있음
- Fragment는 기본적으로 Activity를 본 따 만들었기 때문에 비슷한 구조와 특징을 갖고 있음
- Activity간 이동은 System에서 Intent를 통해 이동하듯이, Fragment는 FragmentManager를 통해 이동
- 이 때 Fragment는 Activity 위에서 동작하기 때문에 Intent가 아닌 메소드 호출을 통해 이동하게 됨

Fragment 클래스

- Fragment 다루기
  - FragmentManager를 사용해서 Fragment를 다룰 수 있음
  - Activity에 존재하는 Fragment를 얻기 위해서는 findFragmentById() 메소드(UI가 있는 Fragment일 경우)로 얻을 수 있음
  - 혹은 findFragmentByTag() 메소드(UI가 있는 Fragment나 UI가 없는 Fragment 둘다)로도 얻을 수 있음

Fragment 클래스

- 메소드
  - public final Activity getActivity()
    - 이 Fragment를 포함하는 Activity를 반환함
  - public final FragmentManager getFragmentManager()
    - 이 Fragment를 포함하는 Activity에서 Fragment 객체들과 의사 소통하는 FragmentManager를 반환함
  - public final Fragment getParentFragment()
    - 이 Fragment를 포함하는 부모가 Fragment일 경우 반환함
    - Activity이면 null을 반환함
  - public final int getId()
    - 이 Fragment의 ID를 반환함

Fragment 클래스

- Fragment 클래스의 Sub 클래스
  - DialogFragment
    - Dialog를 관리하는 데 DialogFragment를 사용하면 사용자가 뒤로 Button을 누르거나 화면을 돌리는 등과 같은 수명 주기 Event를 올바르게 처리할 수 있음
    - DialogFragment 클래스를 사용하면 Dialog의 UI를 더 큰 UI에 삽입할 수 있는 구성 요소로 재사용 가능
    - DialogFragment는 Dialog에 비해 생명 주기를 활용한 Programming이 가능하다는 점에서 용이
  - ListFragment
    - ListFragment는 ListView를 Member로 가지면서, ListView에 대한 구현 절차를 ListView에 대한 직접적인 접근이 아닌, ListFragment에서 제공하는 함수를 통해 수행하도록 만든 클래스
  - PreferenceFragment

Fragment 클래스

- Fragment 생성
  - Fragment class는 Activity class와 유사. Activity 처럼 callback method를 가지고 있음
  - 일반적으로 lifecycle method를 implement 함
    - onAttach()
      - Fragment 가 Activity에 연동될 때 호출
    - onCreate()
      - Fragment가 생성될 때 호출
      - 이때 초기화 작업들을 수행
    - onCreateView()
      - Fragment와 연계된 view가 생성될 때 리턴
    - onActivityCreated()
      - activity에서 fragment가 다 생성된 후에 호출
    - onStart()
      - Fragment가 화면에 보여질 때 호출

Fragment 클래스

- Fragment 생성
  - onResume()
    - Fragment와 유저간의 상호작용이 가능
  - onPause()
    - activity가 중지되거나 해서 Fragment와 유저간의 상호작용이 중단
  - onStop()
    - Fragment가 더 이상 보이지 않을 때 호출
  - onDestroyView()
    - View와 연관된 리소스들이 소멸될 때 호출
  - onDestory()
    - 마지막으로 Fragment 상태가 모두 소멸될 때 호출
  - onDetach()
    - Fragment가 Activity와 연관된 모든 것이 끊어짐
    - Fragment의 자원은 모두 사라짐

FragmentManager 클래스

- Activity와 Fragment의 중간에서 서로를 이어주는 역할을 하는 것이 FragmentManger
  - FragmentTransaction
    - Fragment를 추가, 삭제, 교체 등의 작업을 수행할 수 있게 해주며, 행해진 Transaction의 상태를 Fragment Back Stack에 저장할 수 있도록 함
  - Activity와의 통신
    - 단일 Fragment에 대해 세부적인 작업 또한 가능함
    - Fragment 내의 구성 요소들 하나하나에 접근 할 수 있도록 해줌
    - Activity에서 특정 Event가 발생했을 때, Fragment에서 적절한 동작을 할 수 있도록 함

FragmentManager 클래스

- 메소드
  - FragmentTransaction beginTransaction()
    - Fragment를 변경하기 위한 Transaction을 시작함
  - Fragment findFragmentById(int id)
    - ID를 이용해 Fragment 객체를 찾음
  - Fragment findFragmentByTag(String tag)
    - tag 정보를 이용해 Fragment 객체를 찾음
  - boolean executePendingTransactions()
    - Transaction은 commit() 메소드를 호출하면 실행되지만 비 동기(asynchronous) 방식으로 실행되므로 즉시 실행하고 싶다면 이 메소드를 추가로 호출해야 함

FragmentTransaction 클래스

- FragmentTransaction은 Fragment 연산에 관련된 작업 수행
- Fragment 추가, 삭제 및 교체뿐만 아니라 Fragment Back Stack 관리, Fragment 전환, Animation 설정 등 많은 일을 수행함

FragmentTransaction 클래스

- 메소드
  - add(int, Fragment, String)
    - FragmentManager에 Fragment가 추가
    - int : 컨테이너 ID로써 해당 보통 fragment가 보여줄 부모 Layout ID가 사용
    - Fragment : Activity에 더해질 fragment
    - String : tag명이 들어가는데 추후 FragmentManager에서 tag명으로 fragment를 찾아올 수 있음
      - FragmentManager.findFragmentByTag(String)
    - 리소스 낭비/성능 저하
  - remove(Fragment)
    - 해당 fragment가 존재할 경우 제거
    - FragmentManager에서 더 이상 찾을 수 없음

FragmentTransaction 클래스

- 메소드
  - hide(Fragment)
    - 해당 fragment가 존재할 경우 숨김
    - FragmentManager에 관리 되어지며, 추후 show()를 통해 다시 보여줄 수 있음
  - replace(int container, Fragment, String tag)
    - container안에 tag명으로 존재하는 fragment를 remove(Fragment)한 후에 같은 tag명으로 Fragment를 add(int, Fragment, tag)를 호출함
    - 한마디로 container안에 tag명 fragment를 remove 후에 add 함
    - add()를 사용할 경우 중복되어 같은 fragment가 쌓이는 경우가 발생. Replace()를 사용하는 경우 그 부분을 미리 방지 할 수 있음

FragmentTransaction 클래스

- 메소드
  - detach(Fragment)
    - 해당 fragment가 존재할 경우 떼어냄
    - Remove()와 다른 점은 remove()의 경우 FragmentManager안에서 완전히 제거되지만, detach()의 경우 stack에 존재함
    - 추후 attach()를 호출할 경우 다시 보여줄 수 있지만 View 계층에서 다시 생성되기 때문에 remove()와 별 차이가 없어 보임
  - attach(Fragment)
    - 해당 Fragment가 존재할 경우 다시 View 계층도에 붙임
  - 이 모든 행위를 한 후에 꼭 commit()을 해야 함

Fragment 예제

- 다음과 같은 Fragment 예제를 만들어 보자

Fragment 예제

- Fragment를 만들어 사용하는 과정
  - Fragment를 위한 XML Layout 만들기
  - Fragment 클래스 만들기 (클래스 정의)
    - Fragment를 상속하는 Fragment 클래스 정의
  - Fragment 객체 만들기
    - 예) Fragment fragment = new Fragment();
  - Fragment Manager를 참조하여 필요한 작업 실행하기
    - FragmentManager는 Fragment를 다루는 작업을 해주는 객체로 Fragment를 추가, 삭제 또는 교체 등의 작업을 할 수 있게 함
    - 이런 작업들은 Fragment를 변경할 때 Error가 생기면 다시 원상태로 돌릴 수 있어야 하므로 ‘Transation 객체'를 만들어 실행
    - ‘Transaction 객체'는 beginTransaction() 메소드를 호출하면 시작되고, commit() 메소드를 호출하면 종료

Fragment 예제

- 사용자 인터페이스

```
<?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="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="20dp"
        android:gravity="center"
        android:text="Fragment 모음"
        android:background="#CDDC39"/>
```

Fragment 예제

- 사용자 인터페이스

```
<Button
    android:id="@+id/button1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Fragment"/>

<Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Dialog Fragment"/>

<Button
    android:id="@+id/button3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Preference Fragment"/>
```

Fragment 예제

- 사용자 인터페이스

```
    <Button
        android:id="@+id/button4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="List Fragment"/>

    <Button
        android:id="@+id/button5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="WebView Fragment"/>
</LinearLayout>
```

Fragment 예제

- MainActivity.JAVA

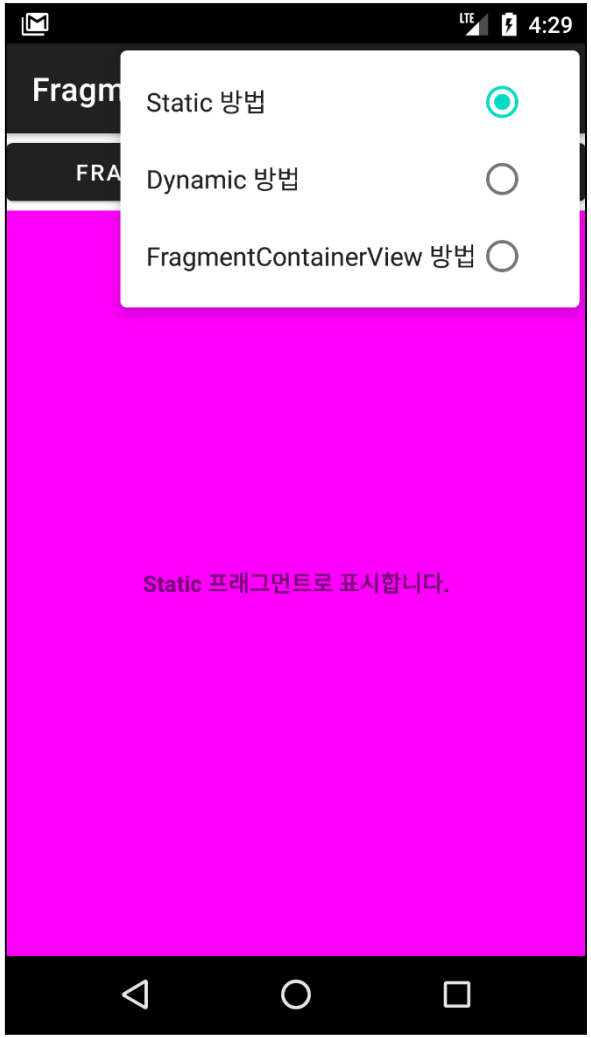
```
public class MainActivity extends AppCompatActivity
                                        implements View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button1 = findViewById(R.id.button1);
        button1.setOnClickListener(this);
        Button button2 = findViewById(R.id.button2);
        button2.setOnClickListener(this);
        Button button3 = findViewById(R.id.button3);
        button3.setOnClickListener(this);
        Button button4 = findViewById(R.id.button4);
        button4.setOnClickListener(this);
        Button button5 = findViewById(R.id.button5);
        button5.setOnClickListener(this);
    }
```

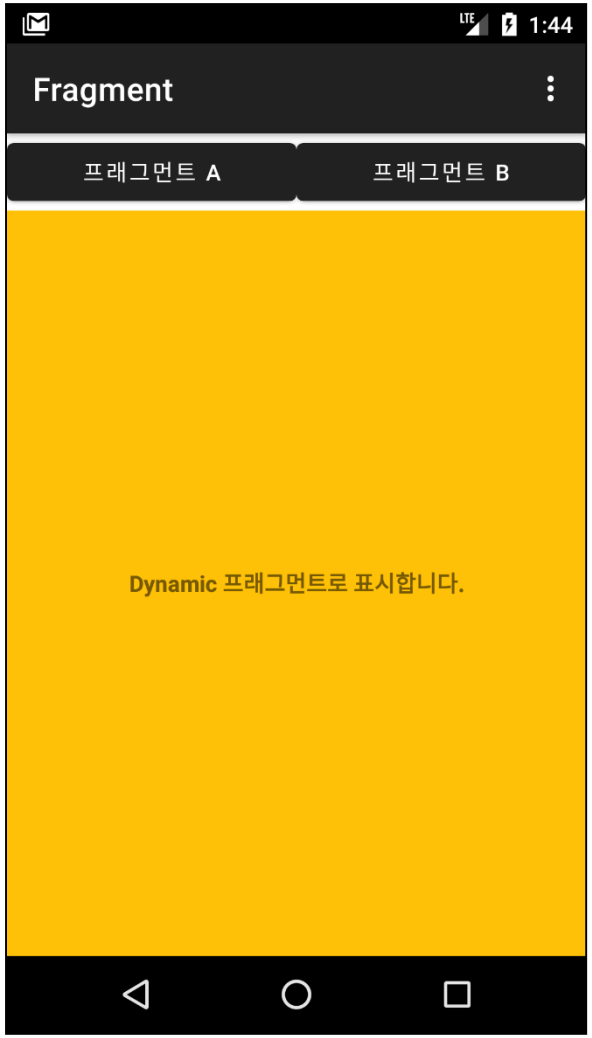
Fragment 예제

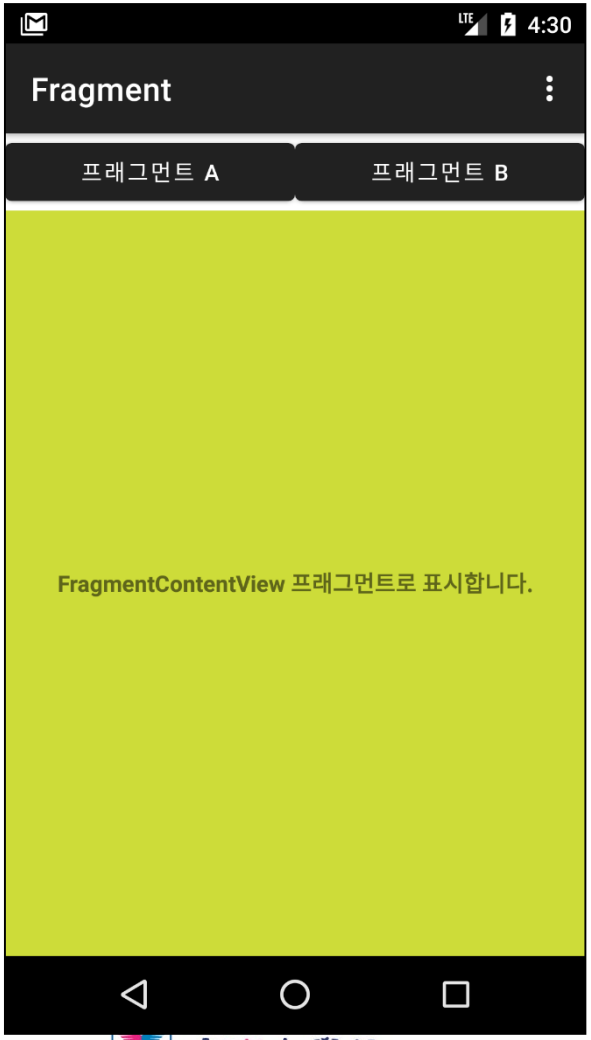
```
    @Override
    public void onClick(View v) {
        Intent intent = null;
        switch (v.getId()) {
            case R.id.button1 :
                intent = new Intent(getBaseContext(), FragmentActivity.class);
                break;
            case R.id.button2 :
                intent = new Intent(getBaseContext(), DialogFragmentActivity.class);
                break;
            case R.id.button3 :
                intent = new Intent(getBaseContext(), PreferenceFragmentActivity.class);
                break;
            case R.id.button4 :
                intent = new Intent(getBaseContext(), ListFragmentActivity.class);
                break;
            case R.id.button5 :
                intent = new Intent(getBaseContext(), WebViewFragmentActivity.class);
        }
        startActivity(intent);
    }
}
```

Fragment 예제(Fragment)

- Fragment

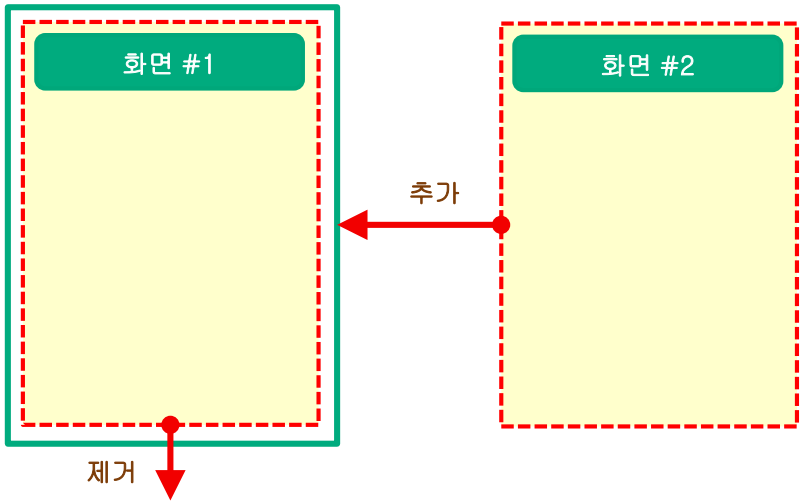






Fragment 예제(Fragment)

- Fragment 전환
  - Fragment Manager와 Transaction을 이용해 추가(add)나 교체(replace) 가능
  - Single Fragment라고 부르며 Activity 전환 없이 화면 전체가 전환되는 효과를 낼 수 있음



Fragment 예제(Fragment)

- 사용자 인터페이스

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

    <LinearLayout
        android:id="@+id/layout1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:visibility="visible">
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
        android:id="@+id/button1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="FragmentA" />

    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="FragmentB" />
</LinearLayout>
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
    <FrameLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <fragment
            android:id="@+id/frame"
            android:name="com.example.fragment.FragmentB"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </FrameLayout>
</LinearLayout>
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
<LinearLayout
    android:id="@+id/layout2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:visibility="invisible">
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="프래그먼트 A" />

    <Button
        android:id="@+id/button4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="프래그먼트 B" />
</LinearLayout>
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
    <FrameLayout
        android:id="@+id/frameLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
<LinearLayout
    android:id="@+id/layout3"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:visibility="invisible">
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="프래그먼트 A" />

    <Button
        android:id="@+id/button6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="프래그먼트 B" />
</LinearLayout>
```

Fragment 예제(Fragment)

- 사용자 인터페이스

```
        <androidx.fragment.app.FragmentContainerView
            android:id="@+id/frameContainer"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </LinearLayout>
</FrameLayout>
```

Fragment 예제(Fragment)

- menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/item1"
            android:checked="true"
            android:title="Static 방법" />
        <item
            android:id="@+id/item2"
            android:title="Dynamic 방법"/>
        <item
            android:id="@+id/item3"
            android:title="FragmentContainerView 방법"    />
    </group>
</menu>
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
public class FragmentActivity extends AppCompatActivity
                                        implements View.OnClickListener {
    private LinearLayout layout1, layout2, layout3;
    private FragmentManager manager;
    private int flag = 1;

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

        layout1 = findViewById(R.id.layout1);
        layout2 = findViewById(R.id.layout2);
        layout3 = findViewById(R.id.layout3);
        manager = getSupportFragmentManager();
    }
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
@Override
protected void onResume() {
    super.onResume();
    Button button1 = findViewById(R.id.button1);
    button1.setOnClickListener(this);
    Button button2 = findViewById(R.id.button2);
    button2.setOnClickListener(this);
    Button button3 = findViewById(R.id.button3);
    button3.setOnClickListener(this);
    Button button4 = findViewById(R.id.button4);
    button4.setOnClickListener(this);
    Button button5 = findViewById(R.id.button5);
    button5.setOnClickListener(this);
    Button button6 = findViewById(R.id.button6);
    button6.setOnClickListener(this);
    button3.performClick();
    button5.performClick();
}
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.item1:
            layout1.setVisibility(View.VISIBLE);
            layout2.setVisibility(View.INVISIBLE);
            layout3.setVisibility(View.INVISIBLE);
            flag = 1;
            break;
        case R.id.item2:
            layout1.setVisibility(View.INVISIBLE);
            layout2.setVisibility(View.VISIBLE);
            layout3.setVisibility(View.INVISIBLE);
            flag = 2;
            break;
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
            case R.id.item3:
                layout1.setVisibility(View.INVISIBLE);
                layout2.setVisibility(View.INVISIBLE);
                layout3.setVisibility(View.VISIBLE);
                flag = 3;
        }
        item.setChecked(true);
        onResume();
        return true;
    }
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
@Override
public void onClick(View v) {
    Fragment fragment = null;
    FragmentTransaction transaction = manager.beginTransaction();
    switch (v.getId()) {
        case R.id.button1:
        case R.id.button3:
        case R.id.button5:
            fragment = new FragmentA(flag);
            break;
        case R.id.button2:
        case R.id.button4:
        case R.id.button6:
            fragment = new FragmentB(flag);
    }
```

Fragment 예제(Fragment)

- FragmentActivity.JAVA

```
        if (v.getId() == R.id.button1 || v.getId() == R.id.button2) {
            transaction.replace(R.id.container, fragment);
            transaction.addToBackStack(null);
        } else if (v.getId() == R.id.button3 || v.getId() == R.id.button4){
            transaction.add(R.id.frameLayout, fragment);
        }else  {
            transaction.add(R.id.frameContainer, fragment);
        }
        transaction.commit();
    }
}
```

Fragment 예제(Fragment)

- fragment_a.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="match_parent"
    android:background="#ffff00"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Static 프래그먼트로 표시합니다."
        android:textStyle="bold" />
</LinearLayout>
```

Fragment 예제(Fragment)

- fragment_aa.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="match_parent"
    android:background="#FFC107"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Dynamic 프래그먼트로 표시합니다."
        android:textStyle="bold" />
</LinearLayout>
```

Fragment 예제(Fragment)

- fragment_aaa.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="match_parent"
    android:background="#CDDC39"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="FragmentContainerView 프래그먼트로 표시합니다."
        android:textStyle="bold" />
</LinearLayout>
```

Fragment 예제(Fragment)

- fragment_b.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="match_parent"
    android:background="#ff00ff"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Static 프래그먼트로 표시합니다."
        android:textStyle="bold" />
</LinearLayout>
```

Fragment 예제(Fragment)

- fragment_bb.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="match_parent"
    android:background="#FF5722"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="Dynamic 프래그먼트로 표시합니다."
        android:textStyle="bold" />
</LinearLayout>
```

Fragment 예제(Fragment)

- fragment_bbb.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="match_parent"
    android:background="#F44336"
    android:orientation="vertical">

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="FragmentContainerView 프래그먼트로 표시합니다."
        android:textStyle="bold" />
</LinearLayout>
```

Fragment 예제(Fragment)

- FragmentA.JAVA

```
public class FragmentA extends Fragment {
    private int flag;

    public FragmentA(int flag) {
        this.flag = flag;
    }
```

Fragment 예제(Fragment)

- FragmentA.JAVA

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view;
    if (flag == 1)
        view = inflater.inflate(R.layout.fragment_a, container, false);
     else if (flag == 2)
        view = inflater.inflate(R.layout.fragment_aa, container, false);
     else
        view = inflater.inflate(R.layout.fragment_aaa, container, false);

     return view;
}
```

Fragment 예제(Fragment)

- FragmentA.JAVA

```
  @Override
    public void onViewCreated(@NonNull View view,
                                                @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        TextView textView = view.findViewById(R.id.text);
        textView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(getContext(), "확인하였습니다",
                                                Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Fragment 예제(Fragment)

- FragmentB.JAVA

```
public class FragmentB extends Fragment {
    private int flag;

    public FragmentB() {
        flag = 1;
    }

    public FragmentB(int flag) {
        this.flag = flag;
    }
```

Fragment 예제(Fragment)

- FragmentB.JAVA

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view;
    if (flag == 1)
     view = inflater.inflate(R.layout.fragment_b, container, false);
    else if (flag == 2)
        view = inflater.inflate(R.layout.fragment_bb, container, false);
    else
        view = inflater.inflate(R.layout.fragment_bbb, container, false);

    return view;
}
```

Fragment 예제(Fragment)

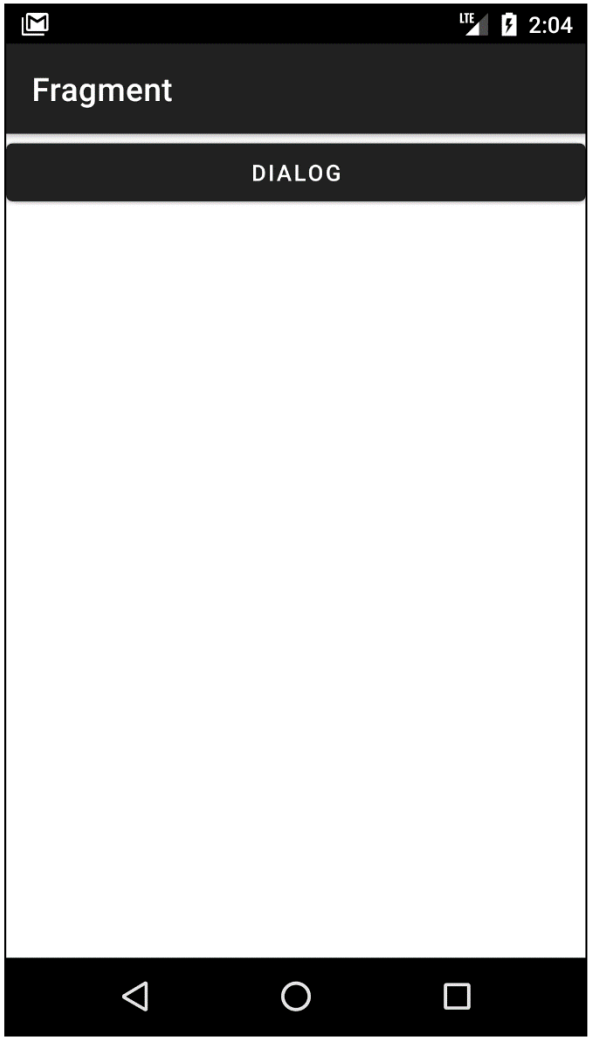
- FragmentB.JAVA

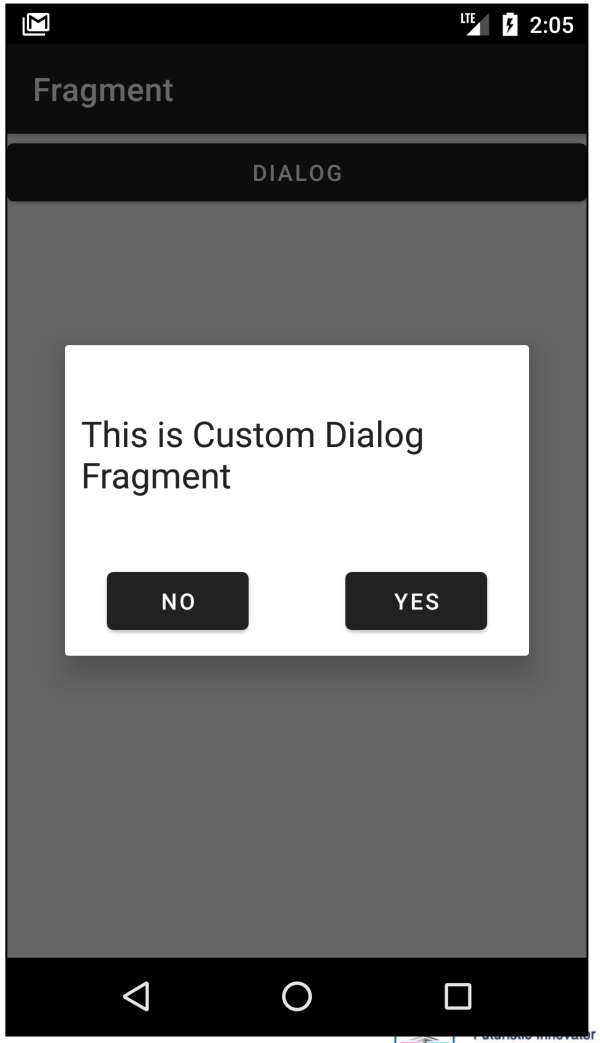
```
    @Override
    public void onViewCreated(@NonNull View view,
                              @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        TextView textView = view.findViewById(R.id.text);
        textView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(getContext(), "확인하였습니다",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

Fragment 예제

- FragmentContainerView
  - Activity UI Layout안에 FragmentContainerView를 정의함으로써 해당 Fragment가 배치될 위치를 정의해야 함
  - FragmentContainerView는 FrameLayout에서 제공하지 않는 조작 관련 사항이 추가된 새로운 View
  - Activity UI Layout 내에 Fragment 자체 UI 위치를 지정할 때는 FragmentContainerView를 사용하길 권장
  - FragmentContainerView가 등장하기 전까지는 Activity에 Fragment를 Hosting할 때 FrameLayout을 많이 사용했음

Fragment 예제(Dialog Fragment)





Fragment 예제(Dialog Fragment)

- activity_dialog_fragment.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="match_parent"
    tools:context=".DialogFragmentActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dialog" />
</LinearLayout>
```

Fragment 예제(Dialog Fragment)

- dialog_fragment.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="match_parent"
    android:orientation="vertical"
    android:padding="10dp">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="30dp"
        android:text="This is Custom Dialog Fragment"
        android:textAppearance="?android:attr/textAppearanceLarge" />
```

Fragment 예제(Dialog Fragment)

- dialog_fragment.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:gravity="center">
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="60dp"
        android:text="No" />
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Yes" />
</LinearLayout>
</LinearLayout>
```

Fragment 예제(Dialog Fragment)

- DialogFragmentActivity.JAVA

```
public class DialogFragmentActivity extends AppCompatActivity {

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

        FragmentManager manager = getSupportFragmentManager();
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                MyDialogFragment fragment = new MyDialogFragment();
                fragment.show(manager, "dialog");
            }
        });
    }
}
```

Fragment 예제(Dialog Fragment)

- DialogFragmentActivity.JAVA

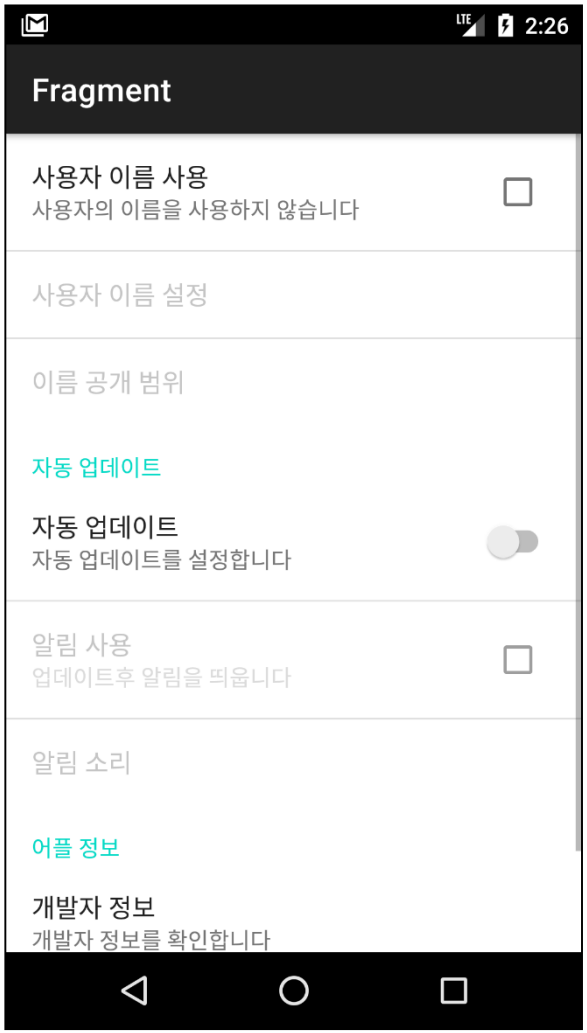
```
public class MyDialogFragment extends DialogFragment
                                    implements View.OnClickListener {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                        Bundle savedInstanceState) {
        setCancelable(false);
        return inflater.inflate(R.layout.dialog_fragment, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view,
                                    @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        Button button1 = view.findViewById(R.id.button1);
        Button button2 = view.findViewById(R.id.button2);
        button1.setOnClickListener(this);
        button2.setOnClickListener(this);
    }
```

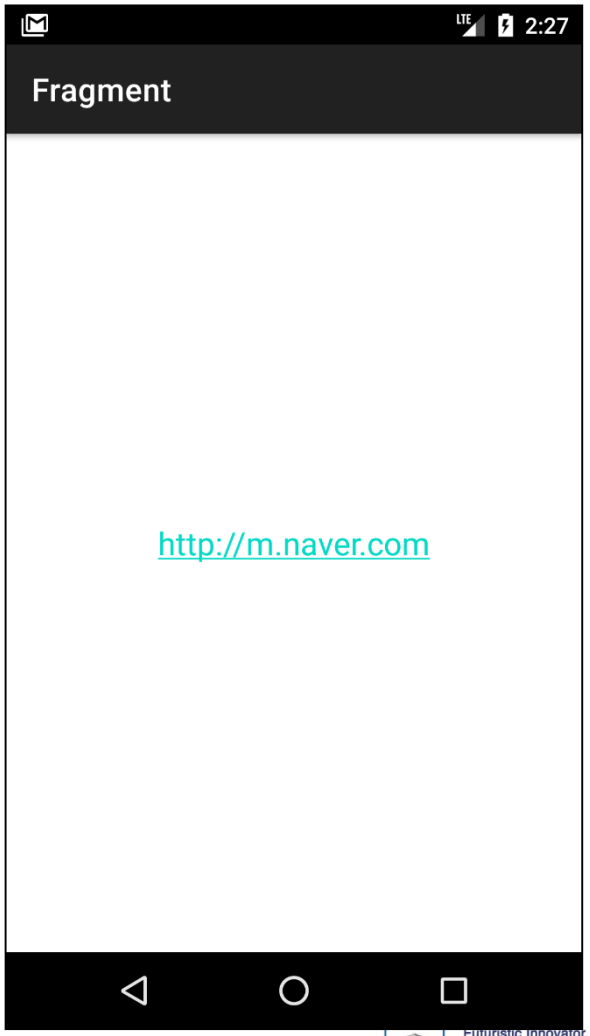
Fragment 예제(Dialog Fragment)

- DialogFragmentActivity.JAVA

```
    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.button2:
                Toast.makeText(getActivity(), "Dialog Yes button clicked",
                                                Toast.LENGTH_SHORT).show();
                break;
            case R.id.button1:
                Toast.makeText(getActivity(), "Dialog No button clicked",
                                                    Toast.LENGTH_SHORT).show();
        }
        dismiss();
    }
}
```

Fragment(Preference Fragment)





Fragment(Preference Fragment)

- activity_preference_fragment.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="match_parent"
    tools:context=".PreferenceFragmentActivity">

    <fragment
        android:id="@+id/fragment"
        android:name="com.example.fragment.MyPreferenceFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
</LinearLayout>
```

Fragment(Preference Fragment)

- PreferenceFragmentActivity.JAVA

```
public class PreferenceFragmentActivity extends AppCompatActivity {

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

Fragment(Preference Fragment)

- pref_settings.xml

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

    <CheckBoxPreference
        android:defaultValue="false"
        android:key="useUserName"
        android:summaryOff="사용자의 이름을 사용하지 않습니다"
        android:summaryOn="사용자의 이름을 사용합니다"
        android:title="사용자 이름 사용" />

    <EditTextPreference
        android:defaultValue="Guest"
        android:dependency="useUserName"
        android:key="userName"
        android:maxLines="1"
        android:selectAllOnFocus="true"
        android:singleLine="true"
        android:title="사용자 이름 설정" />
```

Fragment(Preference Fragment)

- pref_settings.xml

```
<ListPreference
    android:defaultValue="0"
    android:dependency="useUserName"
    android:entries="@array/userNameOpen"
    android:entryValues="@array/userNameOpen_values"
    android:key="userNameOpen"
    android:negativeButtonText="@null"
    android:positiveButtonText="@null"
    android:title="이름 공개 범위" />

<PreferenceCategory android:title="자동 업데이트" >
    <SwitchPreference
        android:defaultValue="false"
        android:key="autoUpdate"
        android:summary="자동 업데이트를 설정합니다"
        android:switchTextOff="OFF"
        android:switchTextOn="ON"
        android:title="자동 업데이트" />
```

Fragment(Preference Fragment)

- pref_settings.xml

```
    <CheckBoxPreference
        android:defaultValue="false"
        android:dependency="autoUpdate"
        android:key="useUpdateNofiti"
        android:summary="업데이트후 알림을 띄웁니다"
        android:title="알림 사용" />

    <RingtonePreference
        android:defaultValue="content://settings/system/notification_sound"
        android:dependency="useUpdateNofiti"
        android:key="autoUpdate_ringtone"
        android:ringtoneType="notification"
        android:showSilent="true"
        android:title="알림 소리" />
</PreferenceCategory>
```

Fragment(Preference Fragment)

- pref_settings.xml

```
<PreferenceCategory android:title="어플 정보" >
    <Preference
        android:summary="개발자 정보를 확인합니다"
        android:title="개발자 정보" >
        <intent
            android:targetClass="com.example.fragment.Developer"
            android:targetPackage="com.example.fragment" />
    </Preference>
    <Preference
        android:summary="개발자에게 메일을 보냅니다"
        android:title="메일 보내기" >
        <intent
            android:action="android.intent.action.SENDTO"
            android:data="mailto:hhbae@kbu.ac.kr" />
    </Preference>
</PreferenceCategory>
</PreferenceScreen>
```

Fragment(Preference Fragment)

- PreferenceFragmenty.JAVA

```
public class PreferenceFragment extends PreferenceFragment {

    public PreferenceFragment() {
    }

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

Fragment(Preference Fragment)

- activity_developer.xml

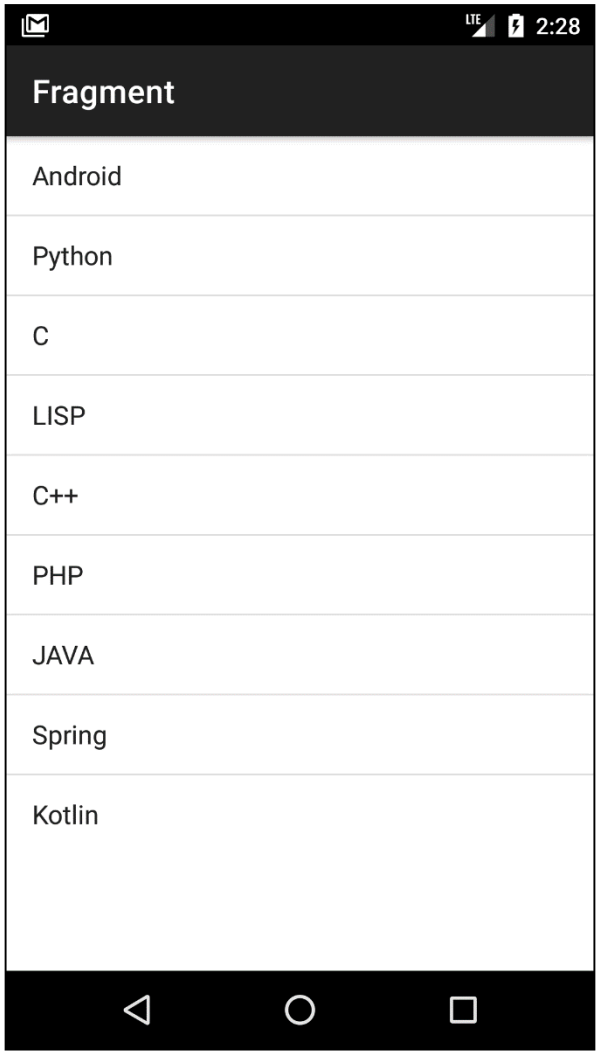
```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="http://m.naver.com"
        android:textSize="20dp"
        android:linksClickable="true"
        android:autoLink="web"/>
</LinearLayout>
```

Fragment(Preference Fragment)

- Developer.JAVA

```
public class Developer extends AppCompatActivity {

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

Fragment(List Fragment)



Fragment(List Fragment)

- activity_list_fragment.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="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/fragment"
        android:name="com.example.fragment.LangListFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Fragment(List Fragment)

- ListFragmentActivity.JAVA

```
public class ListFragmentActivity extends AppCompatActivity {

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

Fragment(List Fragment)

- LangListFragment.JAVA

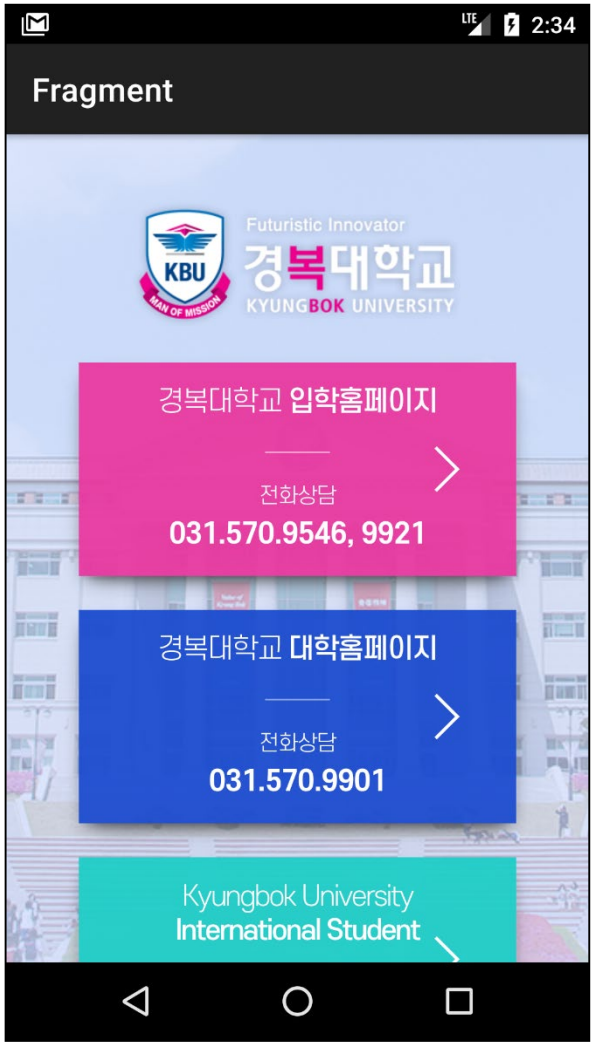
```
public class LangListFragment extends ListFragment {
    String[] language = new String[]{"Android", "Python", "C", "LISP", "C++",
            "PHP", "JAVA", "Spring", "Kotlin"};
    ListView listView;

    @Override
    public void onViewCreated(@NonNull View view,
                              @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(getActivity(),
                    android.R.layout.simple_list_item_activated_1, language);
        FragmentManager manager = getActivity().getSupportFragmentManager();
        LangListFragment listFragment = (LangListFragment)
                                  manager.findFragmentById(R.id.fragment);
        listFragment.setListAdapter(adapter);
        listView = getListView();
        listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    }
```

Fragment(List Fragment)

- LangListFragment.JAVA

```
    @Override
    public void onListItemClick(@NonNull ListView l, @NonNull View v,
                                                    int position, long id) {
        super.onListItemClick(l, v, position, id);
        listView.setItemChecked(position, true);
        Toast.makeText(getActivity(), language[position] +" 선택",
                                                    Toast.LENGTH_SHORT).show();
    }
}
```

Fragment(WebView Fragment)



Fragment(WebView Fragment)

- activity_web_view_fragment.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="match_parent"
    tools:context=".WebViewFragmentActivity">

    <fragment
        android:id="@+id/fragment"
        android:name="com.example.fragment.WebViewFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Fragment(WebView Fragment)

- WebViewFragmentActivity.JAVA

```
public class WebViewFragmentActivity extends AppCompatActivity {

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

Fragment(WebView Fragment)

- web_layout.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="match_parent">

    <WebView
        android:id="@+id/webPage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Fragment(WebView Fragment)

- WebViewFragment.JAVA

```
public class WebViewFragment extends Fragment {
    final String kyungbok = "http://www.kbu.ac.kr";
    String pageurl;

    private class MyWebViewClient extends WebViewClient {
        @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            pageurl = url;
            view.loadUrl(url);
            return true;
        }
    }
```

Fragment(WebView Fragment)

- WebViewFragment.JAVA

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                                        Bundle savedInstanceState) {
        return inflater.inflate(R.layout.web_layout, container, false);
    }

    @Override
    public void onViewCreated(@NonNull View view,
                                                @Nullable Bundle savedInstanceState) {
        WebView webView = view.findViewById(R.id.webPage);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.setWebViewClient(new MyWebViewClient());
        if (pageurl == null) {
            pageurl = kyungbok;
        }
        webView.loadUrl(pageurl);
    }
}
```