8. Fragment

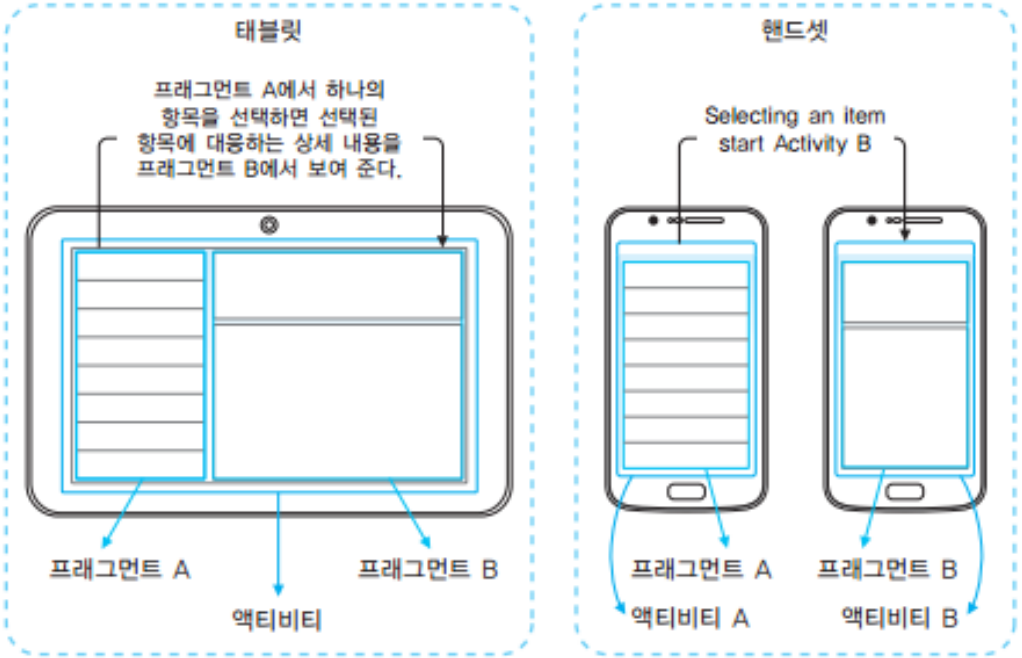
1. 화면 분할 방법 이해하기
2. 프래그먼트 사용하기
3. 프래그먼트 생명주기
4. 동적 프래그먼트
5. 프래그먼트 사이 통신

1. 화면 분할방법 이해하기

개요

- 안드로이드 3.0 이후 추가
- 액티비티안에 위치할 수 있는 UI 조각
- 서브 액티비티(sub Activity)
- 자신만의 생명 주기를 가짐
- 태블릿과 같은 넓은 화면을 가지는 모발일 장치를 위한 메커니즘
- 프래그먼트 클래스

• 여러 개의 패널을 사용하는 동적인 사용자 인터페이스를 만들려면 컴포넌트를 하나의 모듈에 넣어서 이들 모듈을 자유롭게 넣거나 뺄 수 있게 함.

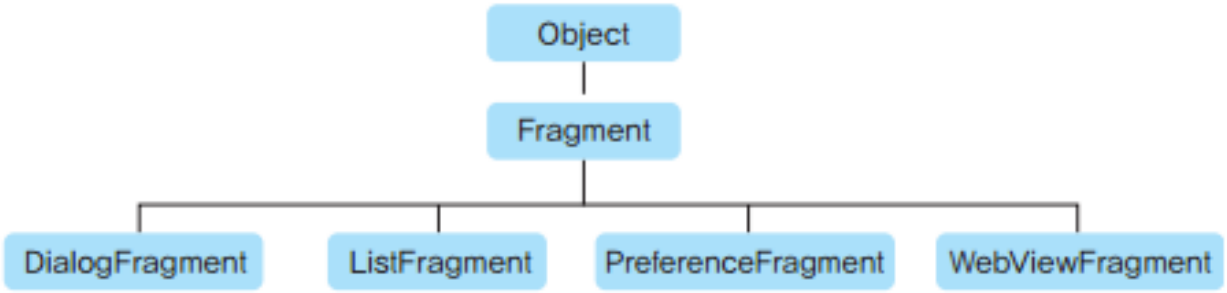


2. 프래그먼트 사용하기

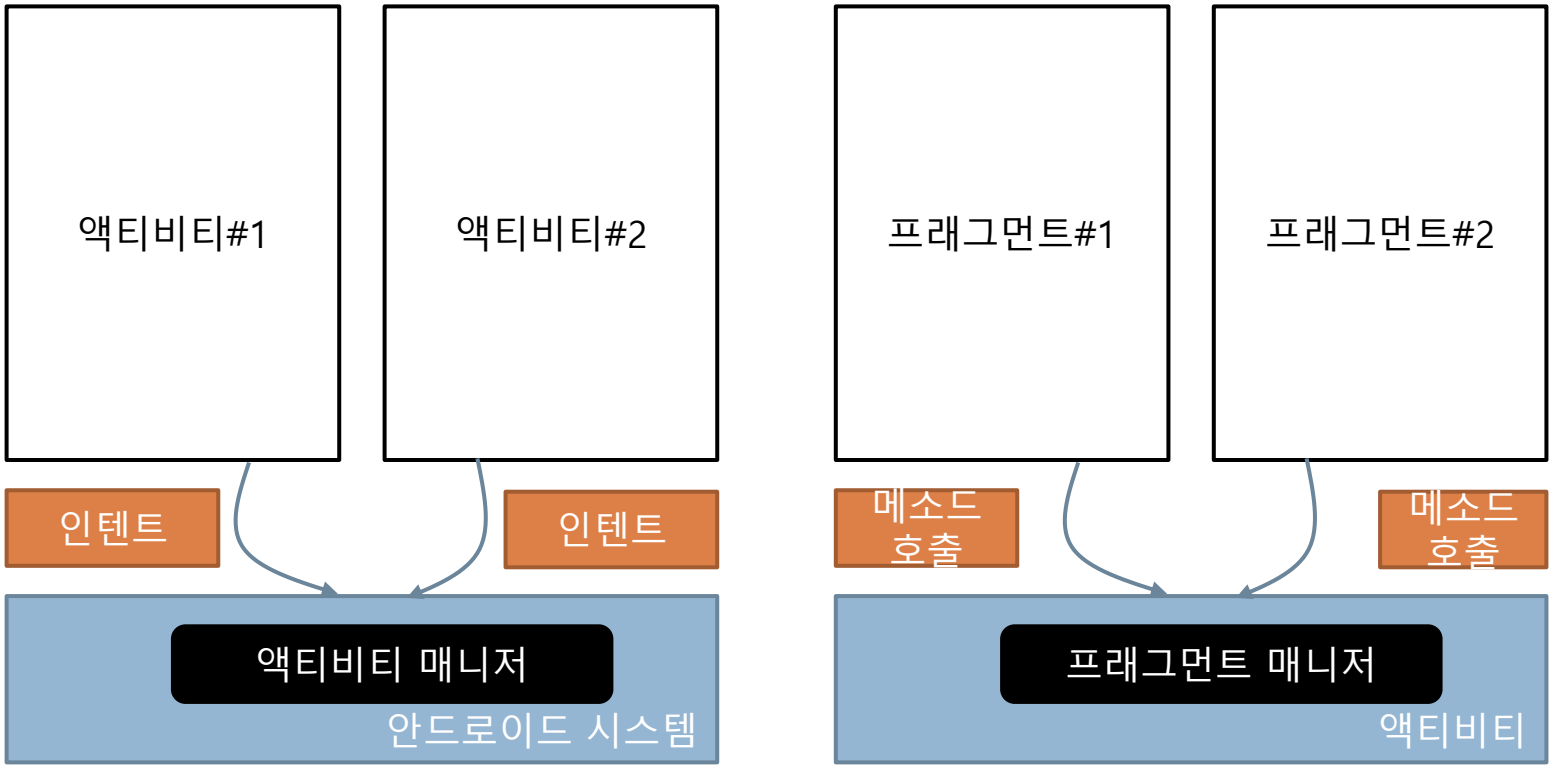
- **프레그먼트 특징**
  - 자신만의 레이아웃을 가짐
  - 자신만의 생명주기를 가짐
  - 액티비티가 실행될 때 프래그먼트를 액티비티에 추가/삭제 가능
  - 하나의 액티비티 안에 여러 개의 프래그먼트를 결합하여서 Multi-pane UI 구현 가능
  - 하나의 프래그먼트를 여러 액티비티에서 사용 가능
  - 프래그먼트 생명 주기는 호스트 액티비티의 생명주기와 연관, 액티비티가 정지되면 그 안의 프래그먼트도 정지

프래그먼트의 계층구조

- 프래그먼트를 생성하려면 Fragment 클래스를 상속한 파생 클래스를 구현
- 안드로이드 시스템은 [용도에 따라 사용할 수 있도록 Fragment 클래스에 대하여 4개의 서브클래스 제공
  - 다이어로그를 위한 DialogFragment,
  - 리스트를 위한 ListFragment,
  - 환경 설정을 위한 PreferenceFragment,
  - 웹뷰을 위한 WebVeiwFragment .



2. 프래그먼트 사용하기


액티비티#1
액티비티#2
인텐트
인텐트
액티비티 매니저
안드로이드 시스템
프래그먼트#1
프래그먼트#2
메소드 호출
메소드 호출
프래그먼트 매니저
액티비티

2. 프래그먼트 사용하기

- **프래그먼트 화면에 추가 방법 이해**

2. 프래그먼트 사용하기

- **Fragment 클래스의 주요 메소드**
  - public final Activity getActivity()
    - 프래그먼트를 포함한 액티비티 리턴
  - public final FragmentManager getFragmentManager();
    - 프래그먼트를 포함하는 포함하는 액티비티에서 객체들과 통신하는 프로그먼트 매니저 리턴
  - public final FragmentManager getSupportFragmentManager();
    - getFragmentManager()와 기능 같음
    - 3.0 이전 버전에서도 호환 가능함
  - public final Fragment getParentFragment();
    - 프래그먼트를 포함하는 부모 프래그먼트 리턴
  - public final int getId();
    - 프래그먼트 ID리턴

2. 프래그먼트 사용하기

- **FragmentManager 클래스**
  - 프래그먼트 관리
  - public abstract FragmentTransaction beginTransaction();
    - 프래그먼트 변경하기 위한 트랙잭션 시작
  - public abstract Fragment findFragmentById(int id);
    - ID를 이용해 프로그먼트 객체를 찾음
  - public abstract Fragment findFragmentByTag(String tag);
    - 태그 정보를 이용해 프래그먼트 객체를 찾음
  - public sbstract boolean ececutePendingTransactions()
    - 트랜잭션을 commit()메소드를 호출하면 실행되지만 비동기 방식으로 실행되므로 즉시 실행하고 싶을때 추가로 호출

3. 프래그먼트 생성하기

프래그먼트 대표적 특성

특성	설명
뷰특성	뷰 그룹에 추가되거나 레이아웃의 일부가 될 수 있다. (뷰에서 상속받는 것은 아니고 뷰를 담고 있는 일종의 틀임)
액티비티 특성	액티비티처럼 수명주기를 가지고 있음 (컨텍스 객체는 아니며 라이프사이클은 액티비티에 종속됨)

프래그먼트 생성하기

- **프로젝트 생성하기**
  - 최소 SKD AP11이상부터 프래그먼트 지원

- **프래그먼트 클래스 생성**

```
public class FragmentA extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // 프래그먼트의 레이아웃을 팽창한다.
        return inflater.inflate(R.layout.fragment_a, container, false);
    }
}
```

프래그먼트 레이아웃 작성: res/layout/fragment_a.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffff00" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="프래그먼트 A입니다."
        android:textStyle="bold" />

</RelativeLayout>
```

프래그먼트를 XML을 이용하여 액티비티에 추가하기

- 플래그먼트는 부모 클래스인 액티비티와 연결되어야 함
- main.xml 파일에서 프래그먼트 정의

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment android:name="kr.co.company.fragmenttest1.FragmentA"
            android:id="@+id/fragmentOne"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />

</LinearLayout>
```

태블릿과 스마트폰에서 화면 다르게 하기

- **FragmentB.java 파일과 res/layout/fragment_b.xml 파일 생성**
  - FragmentB.java 클래스 파일 작성

```
public class FragmentB extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
            ViewGroup container, Bundle savedInstanceState) {

        // 프래그먼트의 레이아웃을 인플레이션한다
        return inflater.inflate(R.layout.fragment_b, container, false);
    }
}
```

- res/layout/fragment_b.xml 파일 작성

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff00ff" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Fragment B"
        android:textSize="12pt"
        android:textStyle="italic" />

</RelativeLayout>
```

- **태블릿을 위한 레이아웃 작성: 태플릿을 위한 res/layout-large폴더 생성, res/layout-large/main.xml**

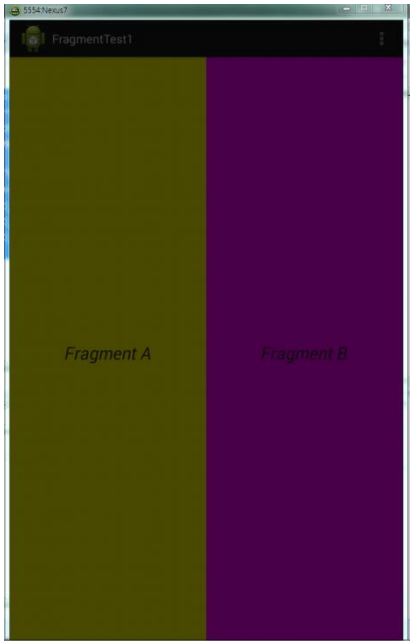
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment android:name="kr.co.company.fragmenttest1.FragmentA"
            android:id="@+id/fragmentOne"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />

    <fragment android:name="kr.co.company.fragmenttest1.FragmentB"
            android:id="@+id/fragmentTwo"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />

</LinearLayout>
```

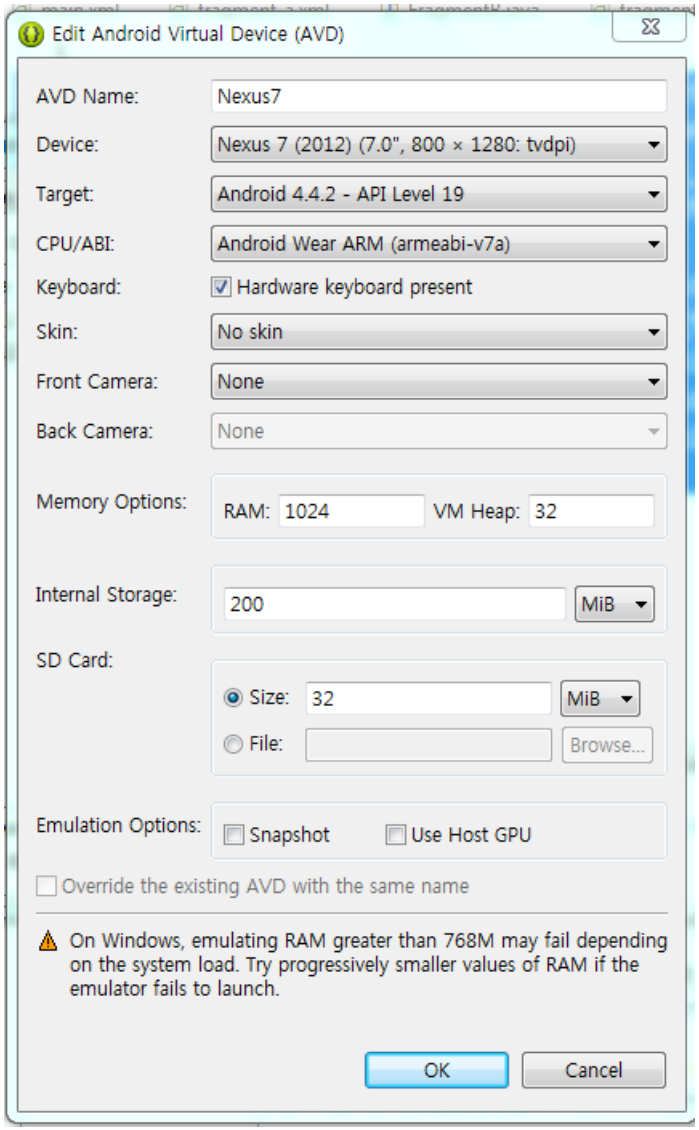
휴대폰을 위한 레이아웃 res/layout/main.xml

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment android:name="kr.co.company.fragmenttest1.FragmentA"
              android:id="@+id/fragmentOne"
              android:layout_weight="1"
              android:layout_width="0dp"
              android:layout_height="match_parent" />

</LinearLayout>
```

테스트를 위해 태블릿 에뮬레이터 생성

- AVD Name: Vexus7
- Device : Nexus7(7.02 800*1200)
- Android 4.1.2
- RAM: 200
- 나머지 설정 동일

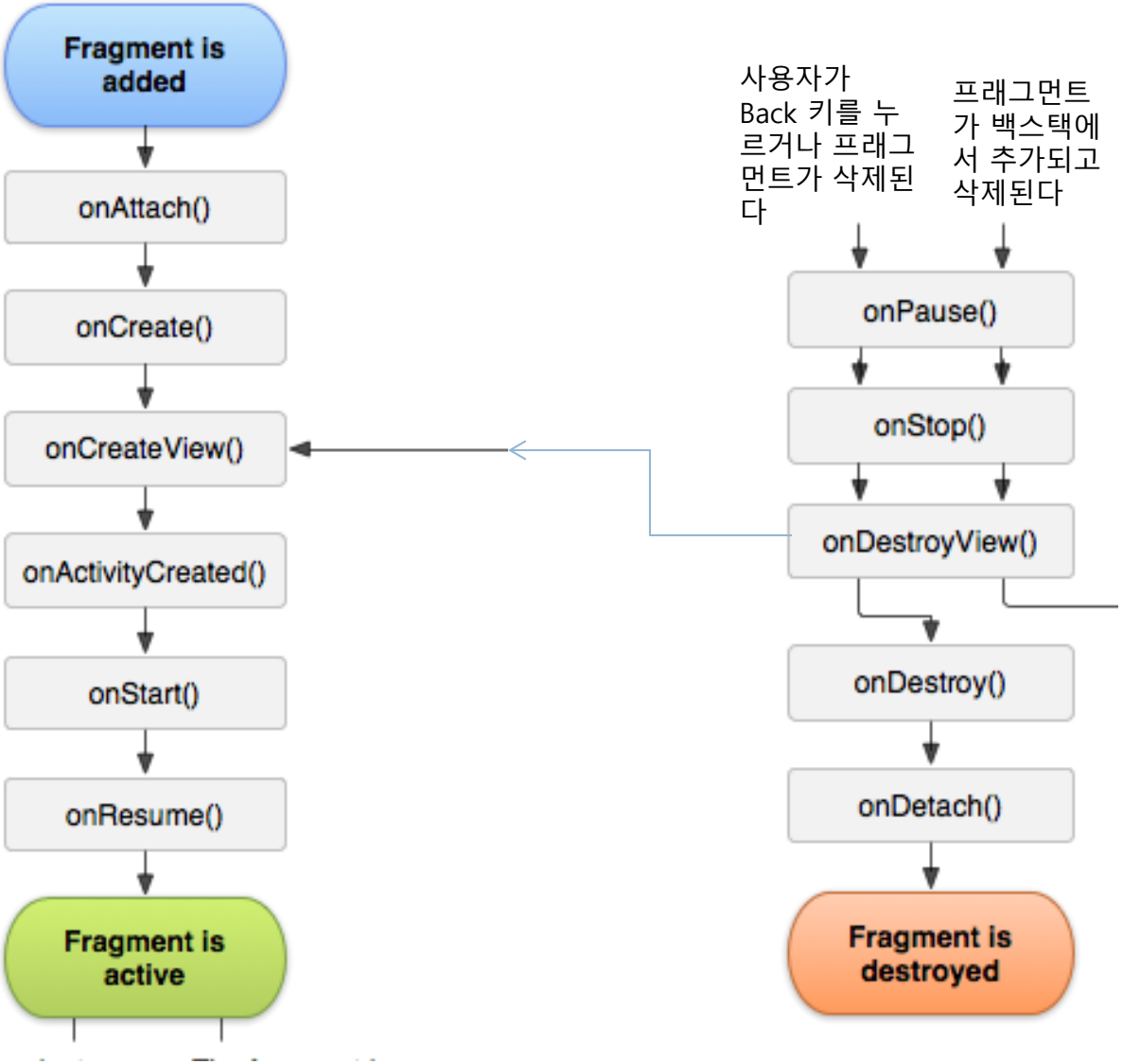




3. 프래그먼트 생명 주기

프로그먼트 생명주기

- 프래그먼트는 액티비티와 같이 상태가 계속 바뀌며, 상태 변화가 발생할 때마다 대응하는 생명주기 콜백 메소드가 호출
- 프래그먼트 3가지 상태
  - 활성resumed 상태 : 실행 중인 액티비티에서 프래그먼트가 보임.
  - 중지paused 상태 : 다른 액티비티가 포그라운드 상태이며 포커스를 가지고 있지만 프래그먼트가 거주하는 액티비티가 여전히 보임.
  - 정지stopped 상태 : 프래그먼트가 보이지 않는다.
    - 호스트 액티비티가 정지되거나 프래그먼트가 액티비티에서 제거되고 백스택에 추가되어 있다.
    - 정지된 프래그먼트는 여전히 살아 있어 모든 상태와 멤버 정보가 시스템에 보관되어 있다.
    - 사용자에게 더 이상 보이지 않고 액티비티가 종료되면 프래그먼트도 종료.


Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
사용자가 Back 키를 누르거나 프래그먼트가 삭제된다
프래그먼트가 백스택에서 추가되고 삭제된다
onPause()
onStop()
onDestroyView()
onDestroy()
onDetach()
Fragment is destroyed

프래그먼트의 생명주기 메소드

- onAttach() : 프래그먼트와 액티비티와 연관될 때 호출.
- onCreate() : 프래그먼트가 생성될 때 호출된다. 프래그먼트가 중지 혹은 정지된 후 재개될 때 보유하기 원하는 프래그먼트의 필수 컴포넌트을 초기화한다.
- onCreateView() : 프래그먼트와 연관된 뷰 계층구조를 생성할 때 호출.
- onActivityCreated() : 프래그먼트를 포함하고 있는 액티비티의 생성이 완료되었을 때, 즉 액티비티의 onCreate() 메서드가 종료될 때 호출.
- onStart() : 프래그먼트가 화면에 표시될 때 호출된다. 아직 사용자와 상호작용은 할 수 없다.
- onResume() : 프래그먼트가 사용자와 상호작용할 수 있을 때 호출.
- onPause() : 프래그먼트가 사용자와 상호작용할 수 없을 때 호출.
- onStop() : 프래그먼트가 화면에서 보이지 않을 때 호출.
- onDestroyView() : 프래그먼트와 연관된 뷰 계층구조가 제거될 때 호출.
- onDestroy() : 프래그먼트가 더 이상 사용될 수 없을 때 호출.
- onDetach() : 프래그먼트가 액티비티와 연동되지 않을 때 호출

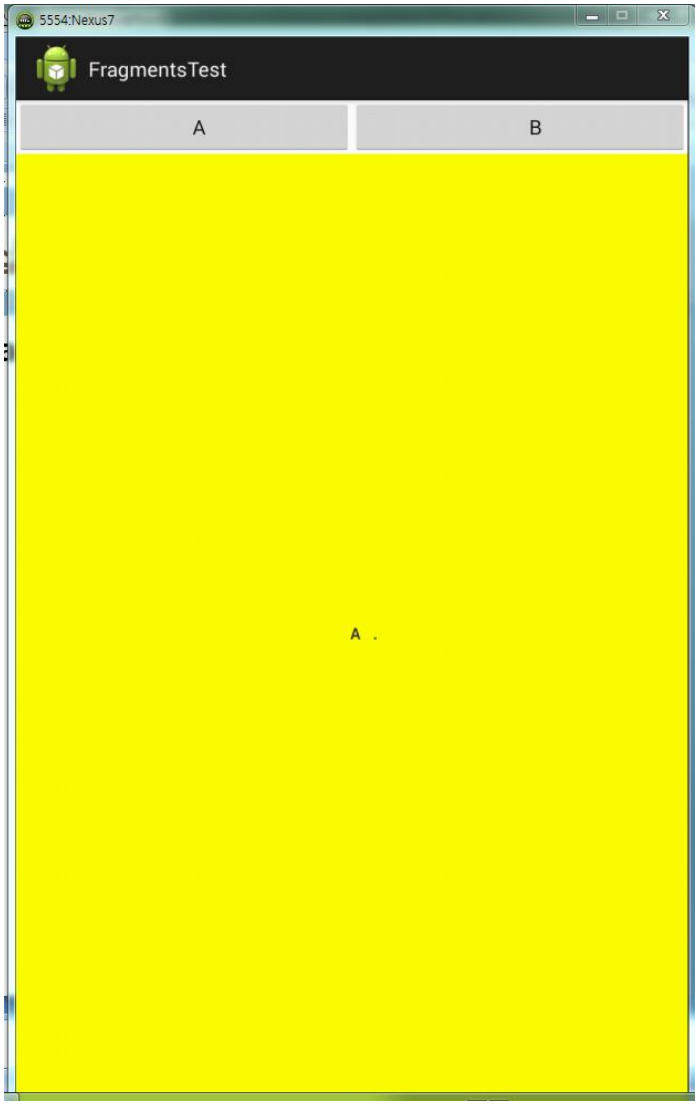
4. 유연한 UI 만들기

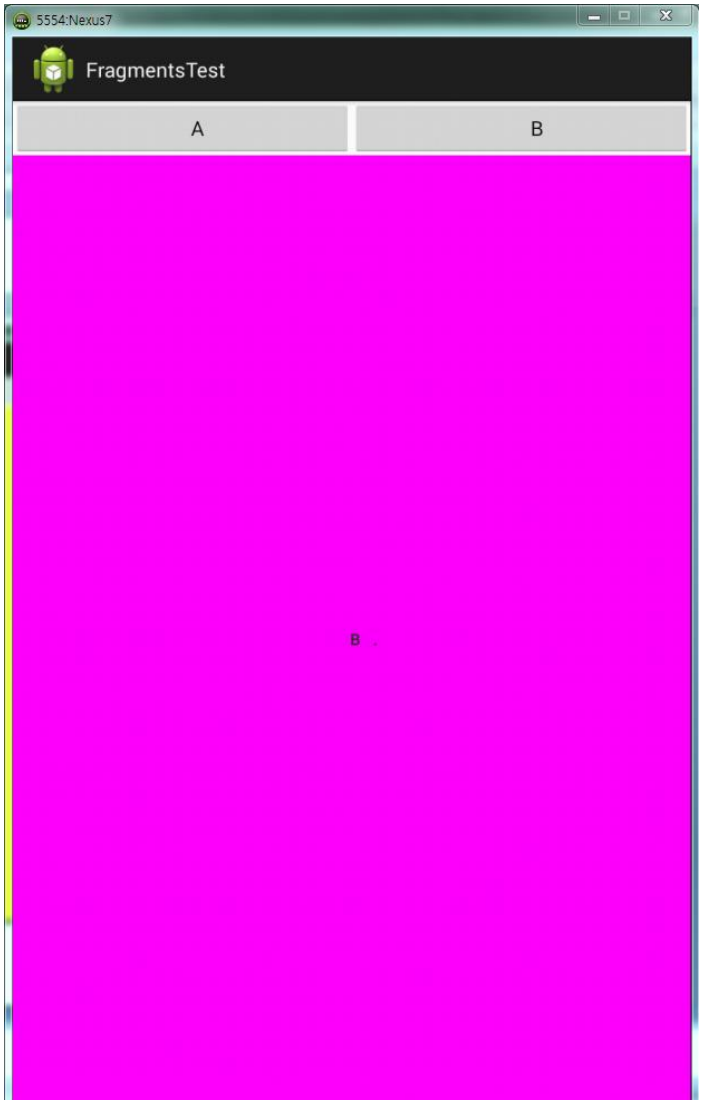
동적 프래그먼트 제어

- FragmentManager 클래스를 이용하여 FragmentTransaction 객체를 생성
- FragmentManager 클래스 : 액티비티에 프래그먼트를 추가, 삭제, 교체하는 메소드를 제공
- 액티비티에 동적으로 프래그먼트를 추가, 제거, 교체하려면 액티비티 onCreate()메소드에 프래그먼트들을 코드로 추가

```
FragmentManager fm = getFragmentManager();
FragmentTransaction fragmentTransaction = fm.beginTransaction();
fragmentTransaction.replace(R.id.fragment_container, fr);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

예제


5554:Nexus7
FragmentsTest
A
B
A


5554:Nexus7
FragmentsTest
A
B
B

res/layout/fragment_a.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffff00" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="프래그먼트 A입니다."
        android:textStyle="bold" />

</RelativeLayout>
```

res/layout/fragment_b.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff00ff" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="프래그먼트 B입니다."
        android:textStyle="bold" />

</RelativeLayout>
```

FragmentA.java, FragmentB.java

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

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

res/layout/main.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" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="selectFragment"
            android:text="프래그먼트 A" />
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:onClick="selectFragment"
            android:text="프래그먼트 B" />
    </LinearLayout>
```

```
        <LinearLayout
            android:id="@+id/fragment_container"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:orientation="horizontal" />

</LinearLayout>
```

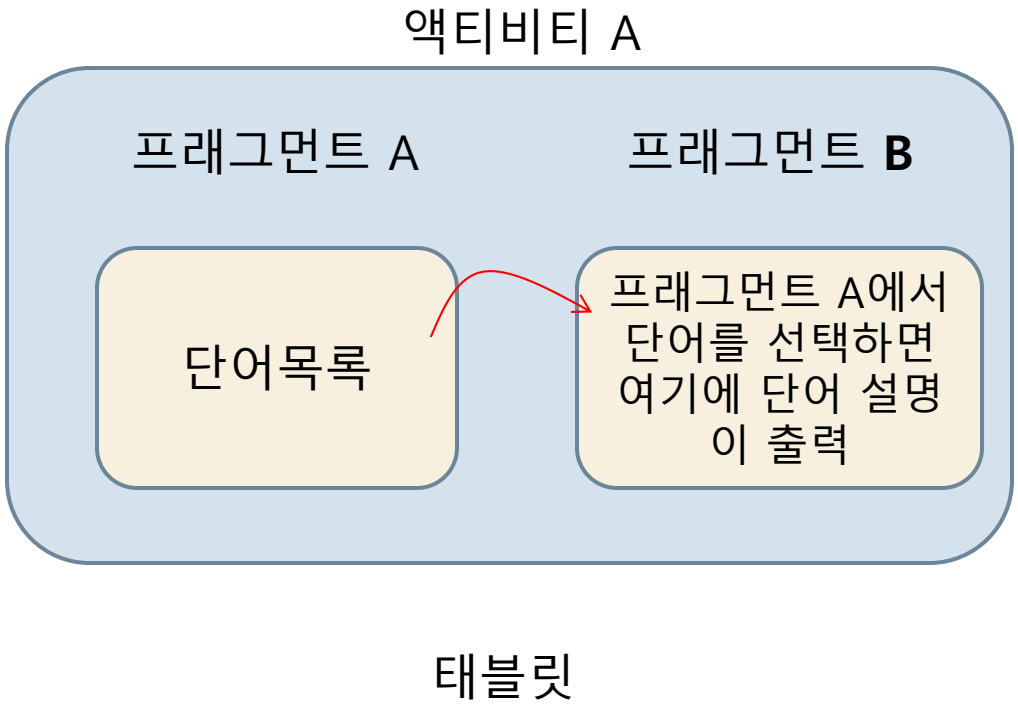
MainActivity.java

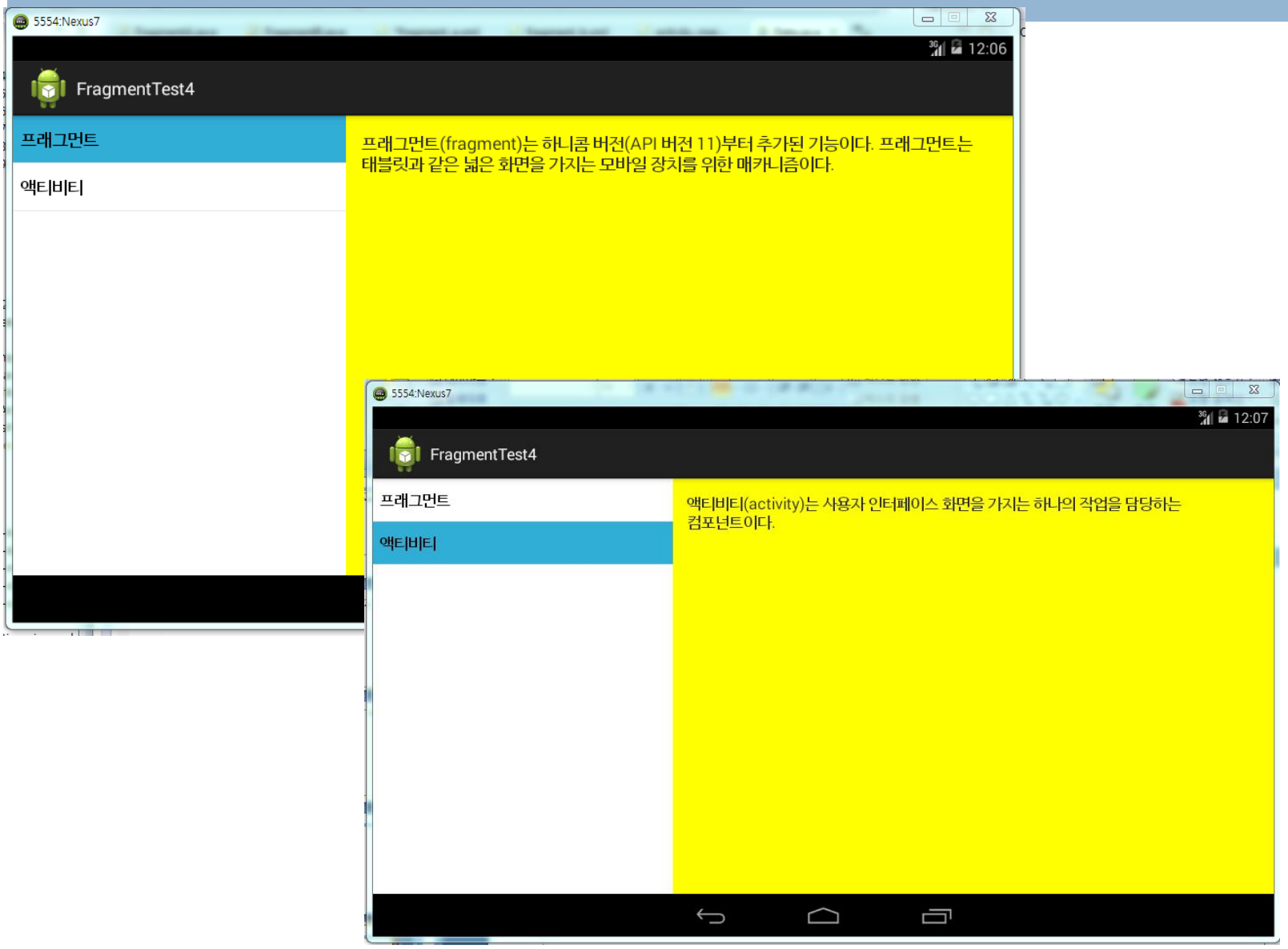
```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if (findViewById(R.id.fragment_container) != null) {
            if (savedInstanceState != null) {
                return;
            }
            FragmentA firstFragment = new FragmentA();
            getFragmentManager().beginTransaction()
                .add(R.id.fragment_container, firstFragment).commit();
        }
    }
    public void selectFragment(View view) {
        Fragment fr = null;
        switch (view.getId()) {
        case R.id.button1: fr = new FragmentA(); break;
        case R.id.button2: fr = new FragmentB(); break;
        }
        FragmentManager fm = getFragmentManager();
        FragmentTransaction fragmentTransaction = fm.beginTransaction();
        fragmentTransaction.replace(R.id.fragment_container, fr);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    }
}
```

5. 다른 프로그먼트와 통신

프래그먼트 사이 통신

- 액티비트를 통해 이루어짐




5554:Nexus7
12:06
FragmentTest4
프래그먼트
액티비티
프래그먼트(fragment)는 하니콤 버전(API 버전 11)부터 추가된 기능이다. 프래그먼트는 태블릿과 같은 넓은 화면을 가지는 모바일 장치를 위한 매카니즘이다.
5554:Nexus7
12:07
FragmentTest4
프래그먼트
액티비티
액티비티(activity)는 사용자 인터페이스 화면을 가지는 하나의 작업을 담당하는 컴포넌트이다.

Data.java

```
public class Data {

    static String[] words = { "프래그먼트", "액티비티" };

    static String[] definitions = {
    "프래그먼트(fragment)는 하니콤 버전(API 버전 11)부터 추가된 기능이다.
    프래그먼트는 태블릿과 같은 넓은 화면을 가지는 모바일 장치를 위한 매카
    니즘이다.",

    "액티비티(activity)는 사용자 인터페이스 화면을 가지는 하나의 작업을 담당
    하는 컴포넌트이다." };
}
```

WordFragment.java

```
public class WordsFragment extends ListFragment {
    OnWordSelectedListener mCallback;

    public interface OnWordSelectedListener {
        public void onWordSelected(int position);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        int layout = Build.VERSION.SDK_INT >=Build.VERSION_CODES.HONEYCOMB ?
        android.R.layout.simple_list_item_activated_1 : android.R.layout.simple_list_item_1;

        setListAdapter(new ArrayAdapter<String>(getActivity(), layout, Data.words));
}
```

```
    @Override
    public void onStart() {
        super.onStart();

        if (getFragmentManager().findFragmentById(R.id.definition_fragment) != null) {
            getListView().setChoiceMode(ListView.CHOICE_MODE_SINGLE);
        }
    }
    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
        try {
            mCallback = (OnWordSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
            + " must implement OnWordSelectedListener");
        }
    }
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        mCallback.onWordSelected(position);
        getListView().setItemChecked(position, true);
    }
}
```

DefinitionFragment.java

```
public class DefinitionFragment extends Fragment {
    final static String ARG_POSITION = "position";
    int mCurrentPosition = -1;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        if (savedInstanceState != null) {
            mCurrentPosition = savedInstanceState.getInt(ARG_POSITION);
        }
        return inflater.inflate(R.layout.definition_view, container, false);
    }

    @Override
    public void onStart() {
        super.onStart();
        Bundle args = getArguments();
        if (args != null) {
            updateDefinitionView(args.getInt(ARG_POSITION));
        } else if (mCurrentPosition != -1) {
            updateDefinitionView(mCurrentPosition);
        }
    }
```

```
    public void updateDefinitionView(int position) {
        TextView def = (TextView) getActivity().findViewById(R.id.definition);
        def.setText(Data.definitions[position]);
        mCurrentPosition = position;
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putInt(ARG_POSITION, mCurrentPosition);
    }
}
```

res/layout/definition_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/definition"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffff00"
    android:padding="16dp"
    android:textSize="18sp" />
```

res/layout-land/main.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="horizontal" >

    <fragment
        android:id="@+id/words_fragment"
        android:name="kr.co.company.fragmenttest4.WordsFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1" />

    <fragment
        android:id="@+id/definition_fragment"
        android:name="kr.co.company.fragmenttest4.DefinitionFragment"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="2" />

</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends FragmentActivity implements
    WordsFragment.OnWordSelectedListener {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        if (findViewById(R.id.fragment_container) != null) {
            if (savedInstanceState != null) {
                return;
            }

            WordsFragment firstFragment = new WordsFragment();
            firstFragment.setArguments(getIntent().getExtras());
            getSupportFragmentManager().beginTransaction()
                .add(R.id.fragment_container, firstFragment).commit();
        }
}
```

```
    public void onWordSelected(int position) {
        DefinitionFragment defFrag = (DefinitionFragment) getSupportFragmentManager()
            .findFragmentById(R.id.definition_fragment);

        if (defFrag != null) {
            defFrag.updateDefinitionView(position);
        } else {
            DefinitionFragment newFragment = new DefinitionFragment();
            Bundle args = new Bundle();
            args.putInt(DefinitionFragment.ARG_POSITION, position);
            newFragment.setArguments(args);
            FragmentTransaction transaction = getSupportFragmentManager()
                .beginTransaction();

            transaction.replace(R.id.fragment_container, newFragment);
            transaction.addToBackStack(null);

            transaction.commit();
        }
    }
}
```