자바응용SW(앱)개발자양성과정

애플리케이션 구성하기

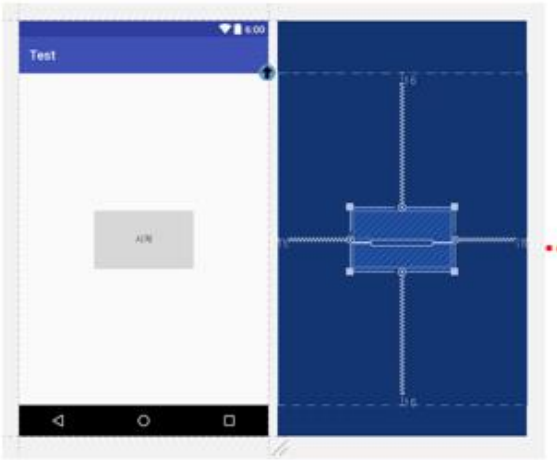
백제직업전문학교

김영준 강사

1. 레이아웃 인플레이션

XML 레이아웃 파일과 자바 소스 파일의 매칭

- setContentView 메소드에서 XML 레이아웃 파일 매칭




```
...
public class MainActivity extends AppCompatActivity {

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

실행

XML 레이아웃 파일

activity_main.xml

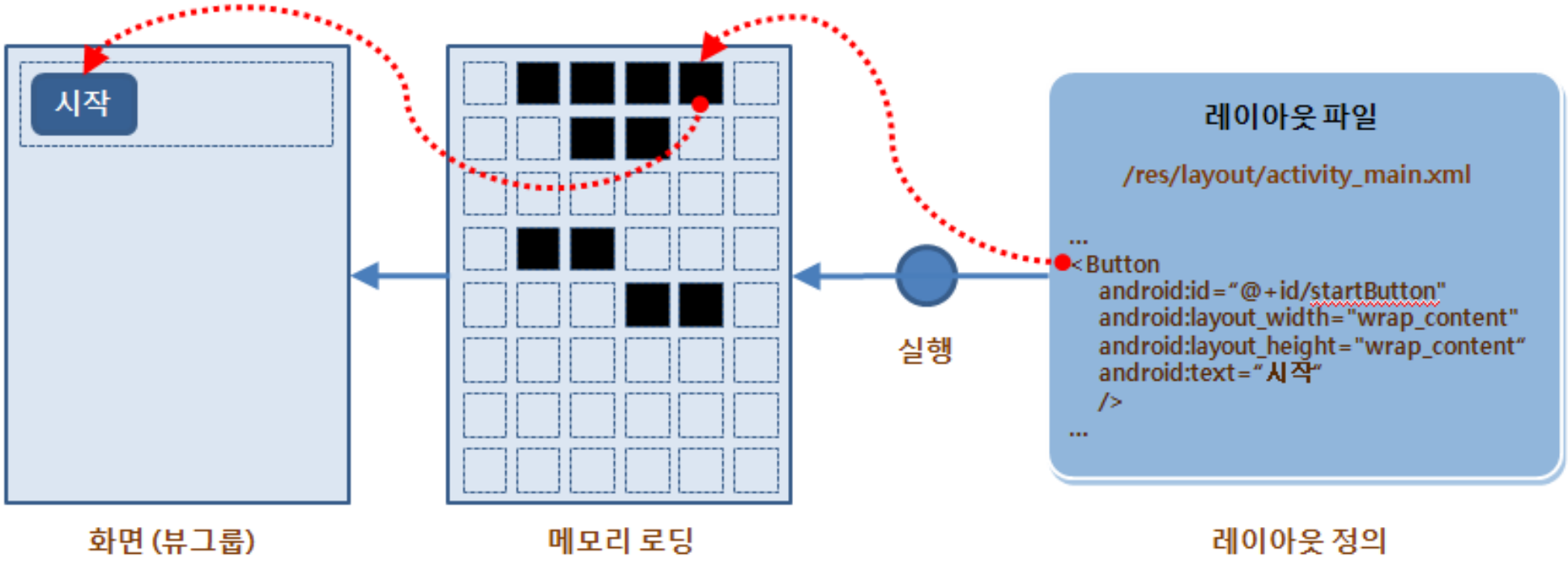
자바 소스 코드 파일

MainActivity.java

R.layout.레이아웃 파일 이름

인플레이션이란?

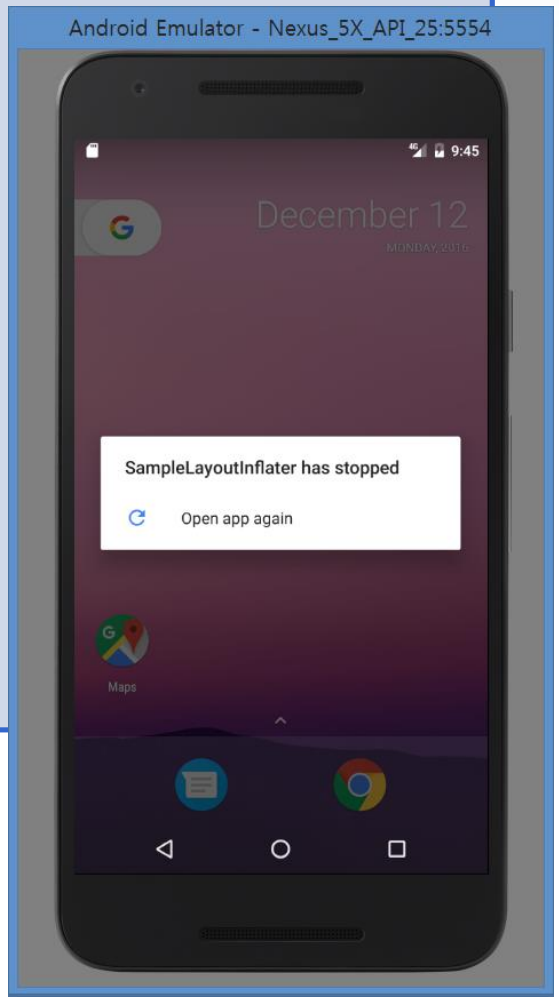
- 인플레이션 : XML 레이아웃에 정의된 내용이 메모리에 객체화되는 과정



["시작" 버튼의 레이아웃 인플레이션 과정]

레이아웃 인플레이션의 이해 – 호출 순서

```
public class MainActivity extends AppCompatActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Button button1 = (Button) findViewById(R.id.button1);
    button1.setText( " 시작됨 " );

    setContentView(R.layout.activity_main);
  }
}
```



[setContentView() 코드와 findViewById() 메소드의 호출 순서를 바꾼 경우]

setContentView() 메소드의 역할

[Reference]

public void setContentView (int layoutResID)
public void setContentView (View view [, ViewGroup.LayoutParams params])

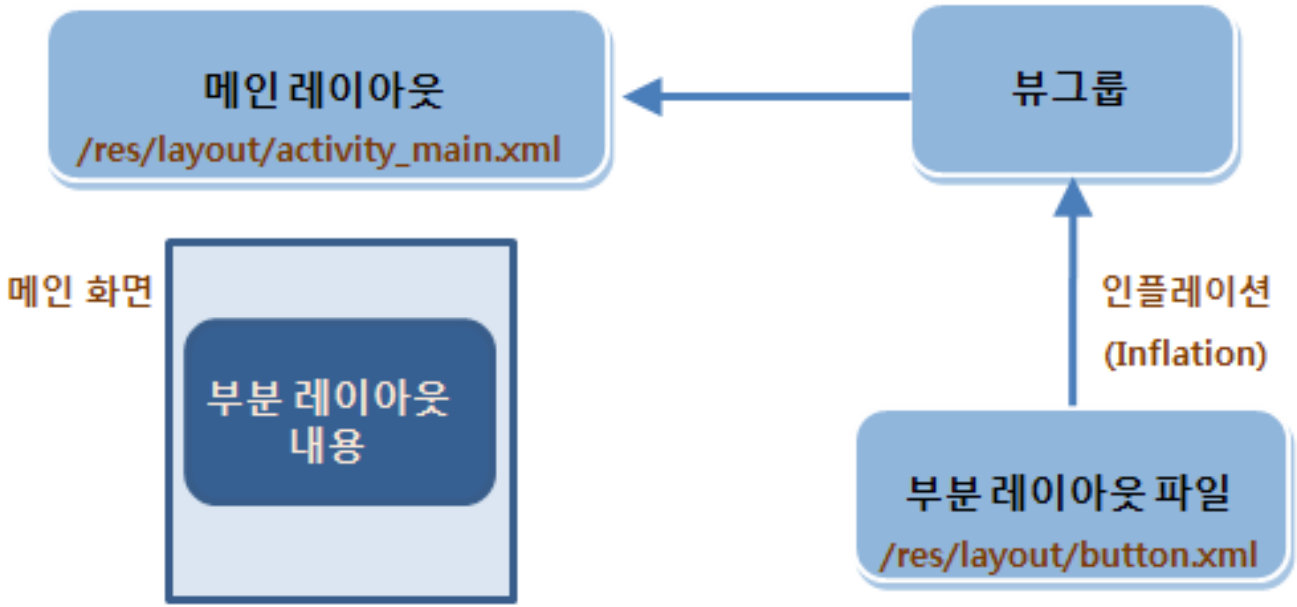
- **setContentView() 메소드의 역할**
  - 화면에 나타낼 뷰를 지정하는 역할
  - XML 레이아웃의 내용을 메모리 상에 객체화하는 역할

[Reference]

getSystemService(Context.LAYOUT_INFLATER_SERVICE)

- **전체 화면 중에서 일부분만을 차지하는 화면 구성요소들을 XML 레이아웃에서 로딩하여 보여줄 수 없을까?**
  - LayoutInflater 라는 클래스를 제공하며, 이 클래스는 시스템 서비스로 제공됨

레이아웃 인플레이션의 개념도



[화면의 일부분을 XML 레이아웃 파일의 내용으로 적용하는 과정]

화면 전체와 화면 일부

- **안드로이드에서 화면 : 소스와 화면 구성이 분리되어 있음**
  - 자바 소스 1개
  - XML 레이아웃 1개

- **화면 전체 : 액티비티 → setContentView 에서 인플레이션**
  - 액티비티를 위한 자바 소스 1개 : MainActivity.java
  - 액티비티를 위한 XML 레이아웃 1개 : activity_main.xml

- **부분 화면 → 수동으로 인플레이션**
  - 부분화면을 위한 자바 소스 1개 또는 뷰 (뷰가 1개의 소스 파일로 분리될 수 있음)
  - 부분화면을 위한 XML 레이아웃 1개 : singer.xml

레이아웃 인플레이션 예제

레이아웃 인플레이션 예제

-화면의 일부로 추가할 뷰의 XML 레이아웃 정의

-레이아웃 인플레이션 후 자바 코드에서 화면의 일부로 추가

메인 액티비티의 XML 레이아웃

-레이아웃 코드 작성

화면 일부의 XML 레이아웃

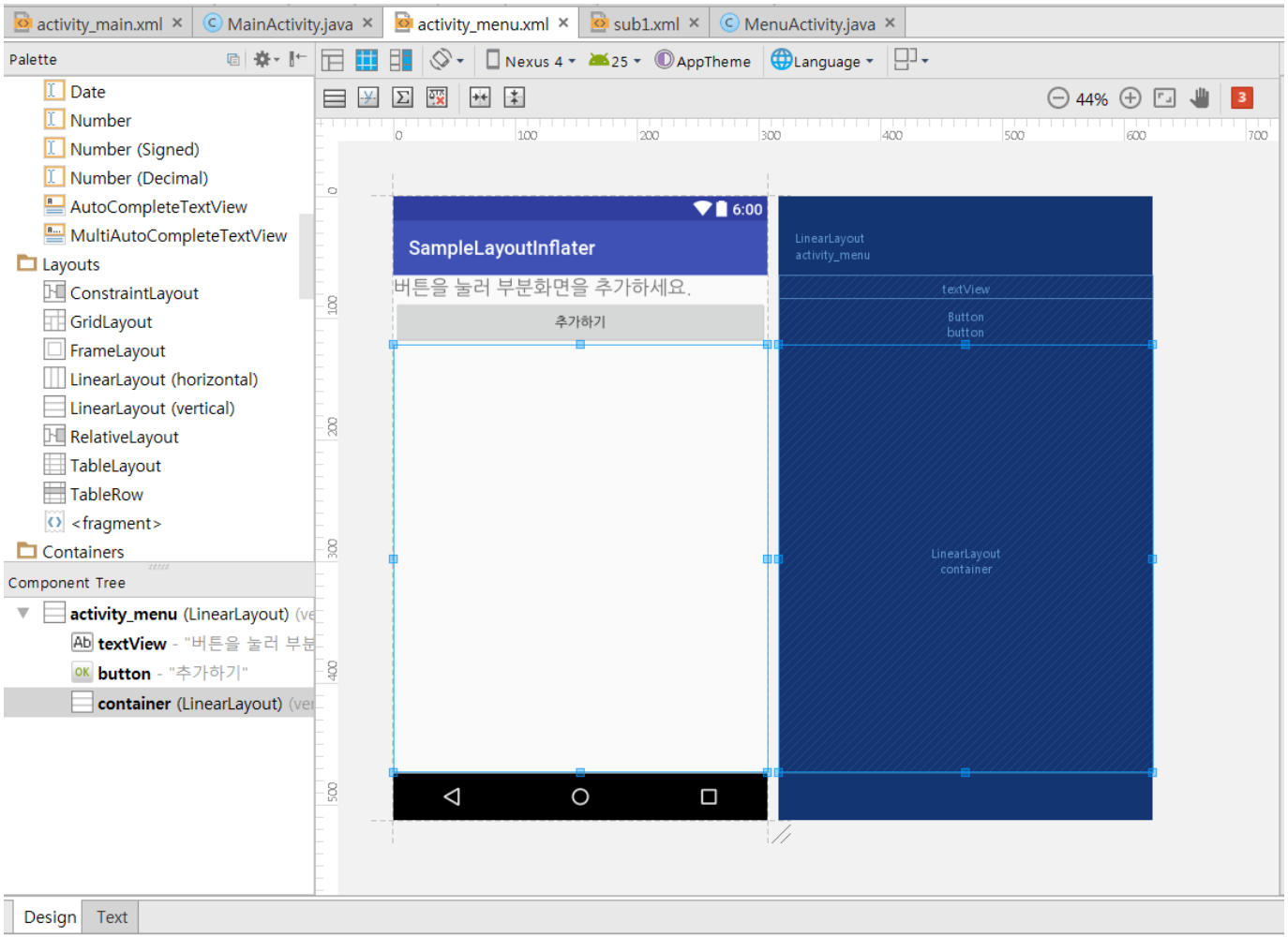
-레이아웃 코드 작성

메인 액티비티 코드

-메인 액티비티 코드 작성

메인 액티비티의 레이아웃

- 위쪽에 버튼을 배치하고 아래쪽에 다른 레이아웃이 들어갈 공간을 확보

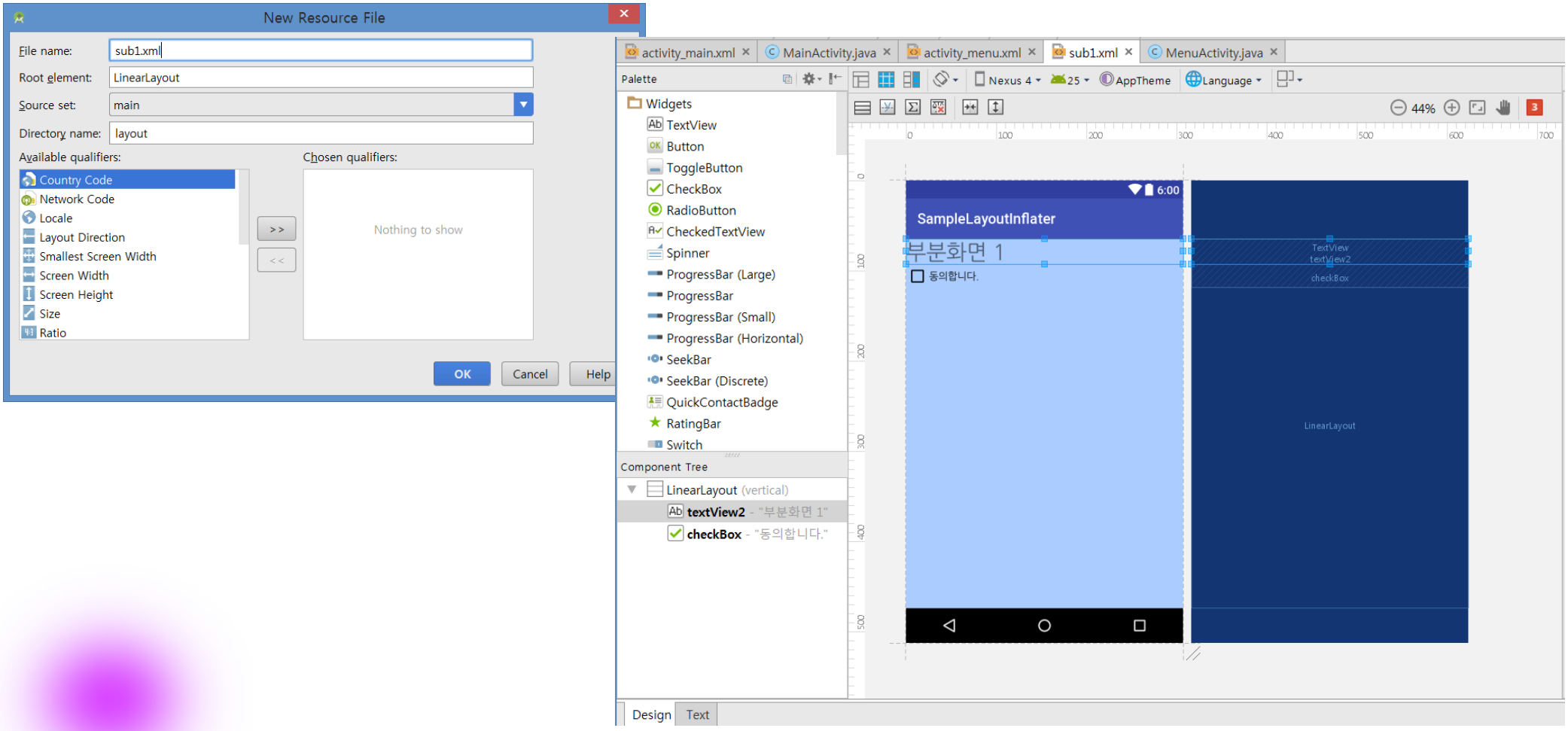


메인 액티비티의 레이아웃

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_menu"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

...

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

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

    </LinearLayout>

</LinearLayout>
```

일부 화면을 위한 레이아웃

- sub1.xml 파일로 만들고 텍스트뷰와 체크박스 위젯 추가한 후 배경색 설정

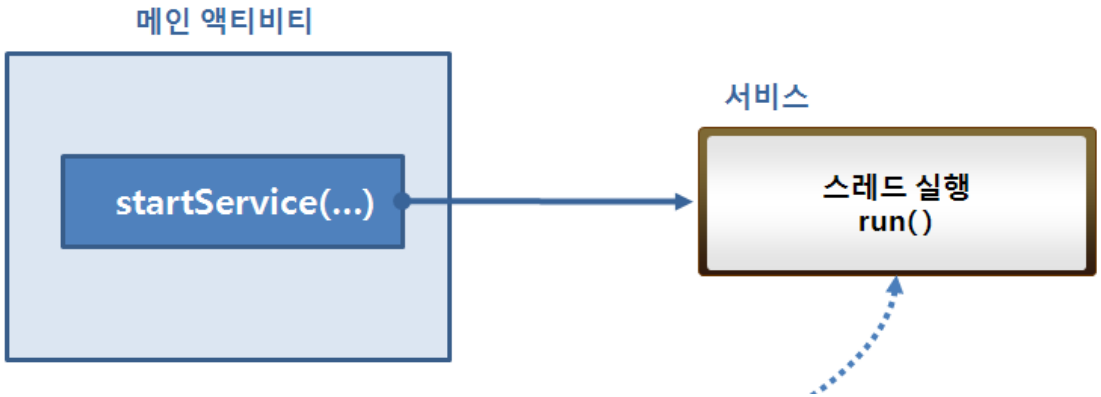


자바 코드 작성

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_menu);

  container = (LinearLayout) findViewById(R.id.container);

  Button button = (Button) findViewById(R.id.button);
  button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        inflater.inflate(R.layout.sub1, container, true);

        CheckBox checkBox = (CheckBox) container.findViewById(R.id.checkBox);
        checkBox.setText("로딩되었어요.");
    }
  });
}
```

5. 서비스

서비스



- 서비스는 **백그라운드**에서 실행되는 애플리케이션 구성 요소
- 서비스는 매니페스트 파일(AndroidManifest.xml) 안에 **<service>** 태그를 이용하여 선언
- 서비스를 시작/중지시키는 메소드
  - Context.startService()
  - Context.bindService()
  - stopService(...)
  - unbindService(...)
- 서비스는 다른 구성 요소들처럼 메인 스레드에서 동작

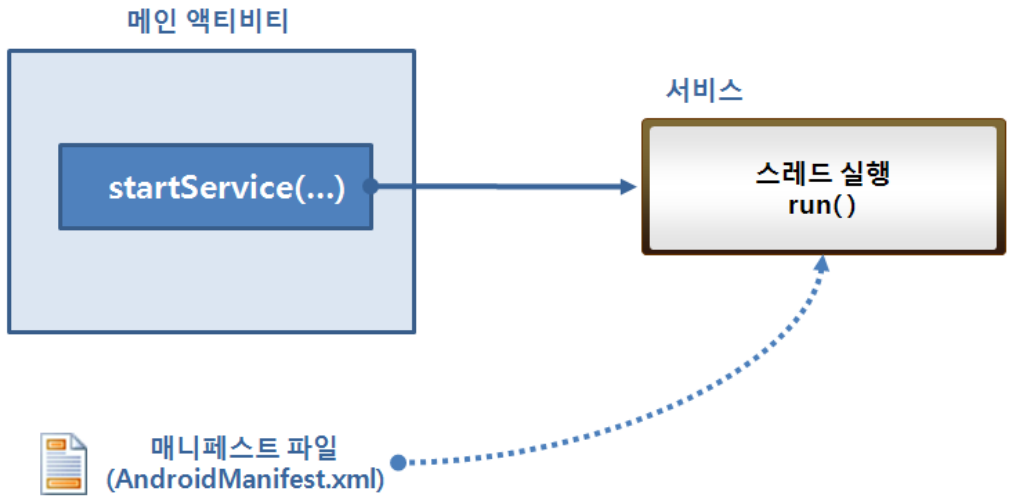
  따라서 CPU를 많이 쓰거나 대기 상태(blocking)를 필요로 하는 작업들은 **스레드를 새로** 만들어 주어야 함

서비스 실행 예제

서비스 예제

-액티비티 상태에 따른 수명주기 확인하기

-상태 메소드 별로 토스트 메시지 추가



서비스 클래스 정의

-일정 시간간격으로 메시지를 보여 주는 서비스 클래스 정의

메인 액티비티 코드 작성

-메인 액티비티에서 서비스 시작

매니페스트에 추가

-새로운 서비스를 매니페스트에 추가

서비스 클래스 정의

```
package org.androidtown.basic.service;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service implements Runnable {
  public static final String TAG = "MyService";
  private int count = 0;
  public void onCreate() {
    super.onCreate();
    Thread myThread = new Thread(this);
    myThread.start();
  }
```

1 시작

Continued..

서비스 클래스 정의 (계속)

```
public void run() {
 while(true) {
  try {
   Log.i(TAG, "my service called #" + count);
   count++;
   Thread.sleep(5000);
  } catch(Exception ex) {
   Log.e(TAG, ex.toString());
   }
   }
  }
  @Override
  public IBinder onBind(Intent arg0) {
  return null;
 }
}
```

2 기록

메인 액티비티 코드 만들기

```
public class MainActivity extends AppCompatActivity {
 @Override
 public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  Intent myIntent = new Intent(this, MyService.class );
  startService(myIntent);
 }
}
```

1 인텐트 객체 생성

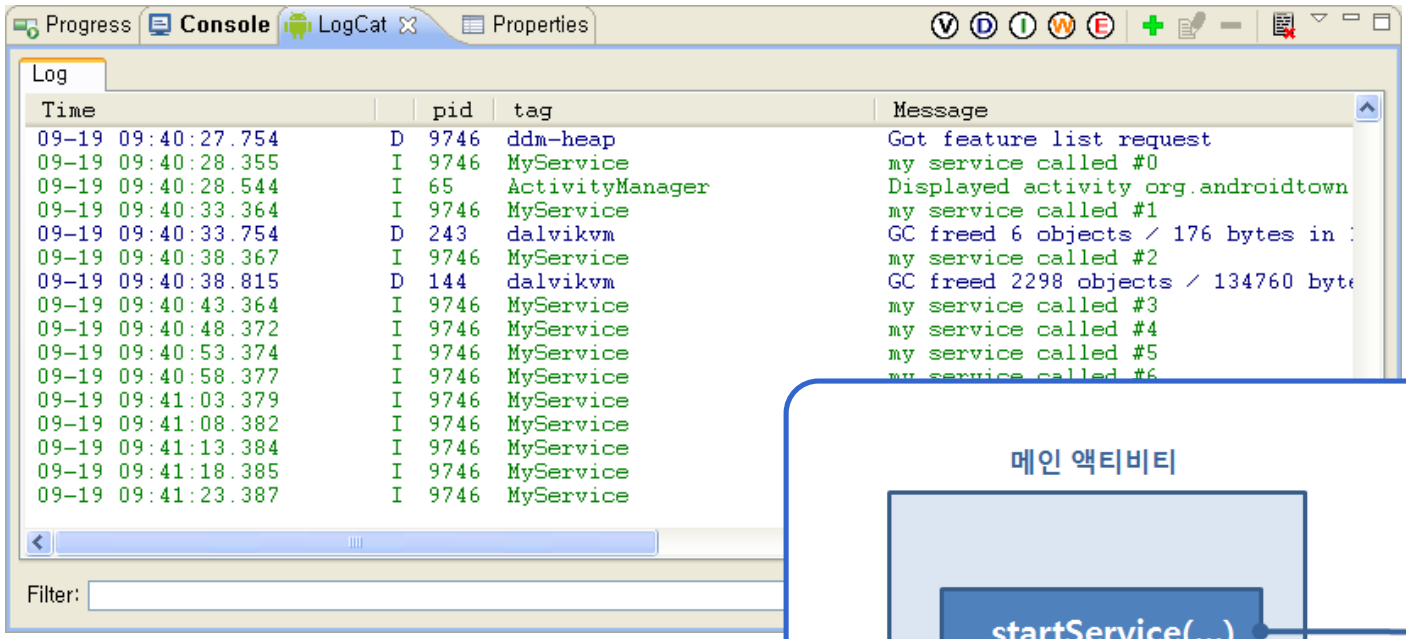
2 서비스 시작

매니페스트에 추가하기

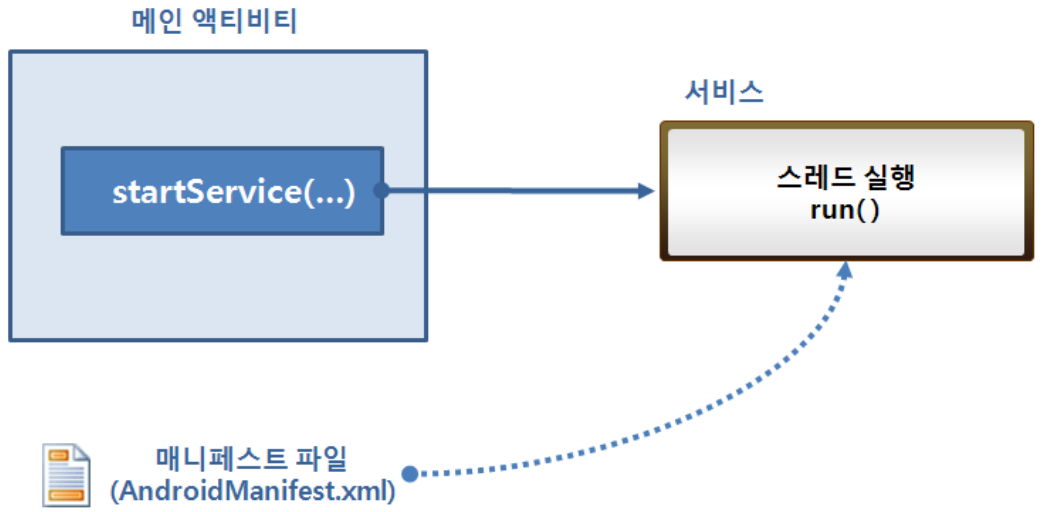
```
...
<application android:icon="@drawable/icon" android:label="@string/app_name">
 ...
  <service android:name="MyService" >
  </service>

</application>
...
```

1 서비스 등록

서비스 실행하여 로그 확인



[서비스를 통해 로그를 남긴 경우]



[서비스 이용 패턴]

서비스 예제

배경에서 음악을 연주하는 서비스를 작성하여 보자.

mp3 형식의 음악 파일을 하나 다운로드 받아서 /res/raw 디렉토리에 old_pop.mp3와 같은 이름으로 저장한다.

음악을 연주하는 서비스

```
public class MusicService extends Service {

        private static final String TAG = "MusicService";
        MediaPlayer player;

        public IBinder onBind(Intent intent) {
                return null;
        }
        public void onCreate() {
                Log.d(TAG, "onCreate()");
                player = MediaPlayer.create(this, R.raw.old_pop);
                player.setLooping(false); // Set looping
        }
```

음악을 연주하는 서비스

```
        public void onDestroy() {
                Toast.makeText(this, "Music Service가 중지되었습니다.",
                                Toast.LENGTH_LONG).show();
                Log.d(TAG, "onDestroy()");
                player.stop();
        }
        public int onStartCommand(Intent intent, int flags, int startId) {
                Toast.makeText(this, "Music Service가 시작되었습니다.",
                                Toast.LENGTH_LONG).show();
                Log.d(TAG, "onStart()");
                player.start();
                return super.onStartCommand(intent, flags, startId);
        }
}
```

매니페스트 파일

```
<manifest
        ...
<application
        <activity          ...          </activity>
        <service
                android:enabled="true"
                android:name=".MusicService“
         />
</application>
</manifest>
```

서비스 사용 예제

앞의 서비스를 사용하는 예제를 작성
먼저 다음과 같은 인터페이스를 XML로 작성

```
<?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="top|center"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:padding="20dp"
        android:text="음악 서비스 테스트"
        android:textSize="20sp" />

    <Button
        android:id="@+id/start"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="시작" >
    </Button>

    <Button
        android:id="@+id/stop"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="중지" >
    </Button>

</LinearLayout>
```

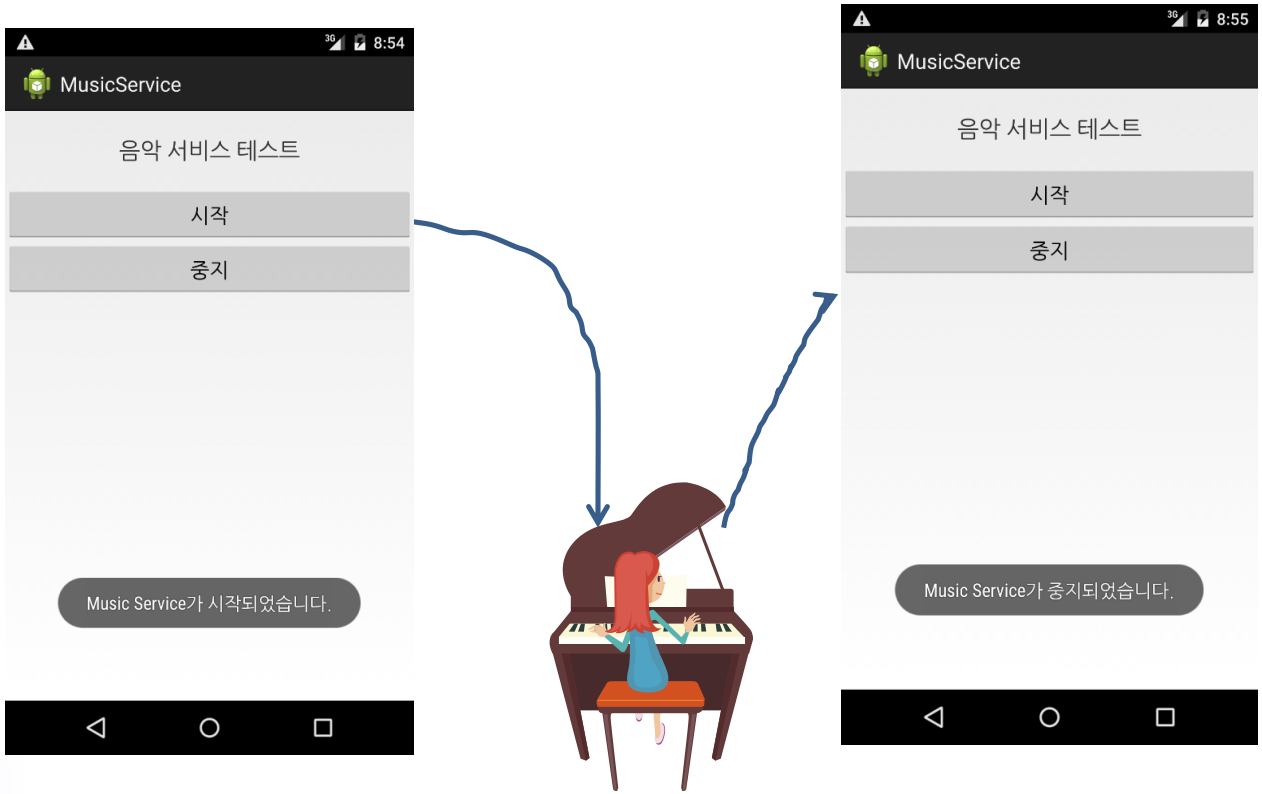
MusicserviceTest.java

```
...
public class MusicServiceTest extends Activity implements OnClickListener  {
  private static final String TAG = "MusicServiceTest";
  Button start, stop;

  @Override
  public void onCreate(Bundle savedInstanceState)  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    start = (Button) findViewById(R.id.start);
    stop = (Button) findViewById(R.id.stop);
    start.setOnClickListener(this);
    stop.setOnClickListener(this);
}
```

MusicserviceTest.java

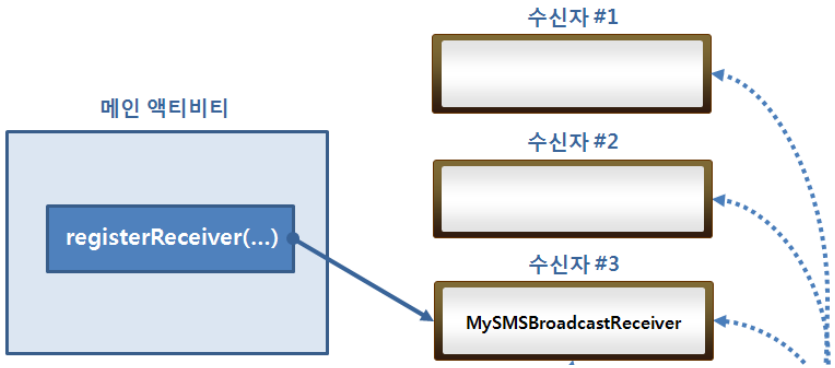
```
public void onClick(View src) {
  switch (src.getId())  {
  case R.id.start:
    Log.d(TAG, "onClick() start ");
    startService(new Intent(this, MusicService.class));
    break;

  case R.id.stop:
    Log.d(TAG, "onClick() stop");
    stopService(new Intent(this, MusicService.class));
    break;
  }
 }
}
```

실행 결과


8:54
MusicService
음악 서비스 테스트
시작
중지
Music Service가 시작되었습니다.
8:55
MusicService
음악 서비스 테스트
시작
중지
Music Service가 중지되었습니다.

6. 브로드캐스트 수신자

브로드캐스트 수신자



- 애플리케이션이 글로벌 이벤트(global event)를 받아서 처리하려면 브로드캐스트 수신자로 등록
- 글로벌 이벤트란 "전화가 왔습니다.", "문자 메시지가 도착했습니다."와 같이 안드로이드 시스템 전체에 보내지는 이벤트
- 브로드캐스트 수신자는 인텐트필터를 포함하며, 매니페스트 파일에 등록함으로써 인텐트를 받을 준비를 함
- 수신자가 매니페스트 파일에 등록되었다면 따로 시작시키지 않아도 됨
- 애플리케이션은 컨텍스트 클래스의 registerReceiver 메소드를 이용하면 런타임 시에도 수신자를 등록할 수 있음
- 서비스처럼 브로드캐스트 수신자도 UI가 없음

브로드캐스트의 구분

- **인텐트와 브로드캐스트**
  - 인텐트를 이용해서 액티비티를 실행하면 포그라운드(foreground)로 실행되어 사용자에게 보여지지만
  - 브로드캐스트를 이용해서 처리하면 백그라운드(background)로 동작하므로 사용자가 모름
  - 인텐트를 받으면 onReceive() 메소드가 자동으로 호출됨

- **브로드캐스트의 구분**

  브로드캐스트는 크게 두 가지 클래스로 구분됨

  - 일반 브로드캐스트 (sendBroadcase() 메소드로 호출)

    비동기적으로 실행되며 모든 수신자는 순서없이 실행됨 (때로는 동시에 실행됨)

    효율적이나, 한 수신자의 처리 결과를 다른 수신자가 이용할 수 없고 중간에 취소불가

  - 순차 브로드캐스트 (sendOrderedBroadcase() 메소드로 호출)

    한 번에 하나의 수신자에만 전달되므로 순서대로 실행됨. 중간에 취소하면 그 다음 수신자는 받지 못함. 수신자가 실행되는 순서는 인텐트필터의 속성으로 정할 수 있음

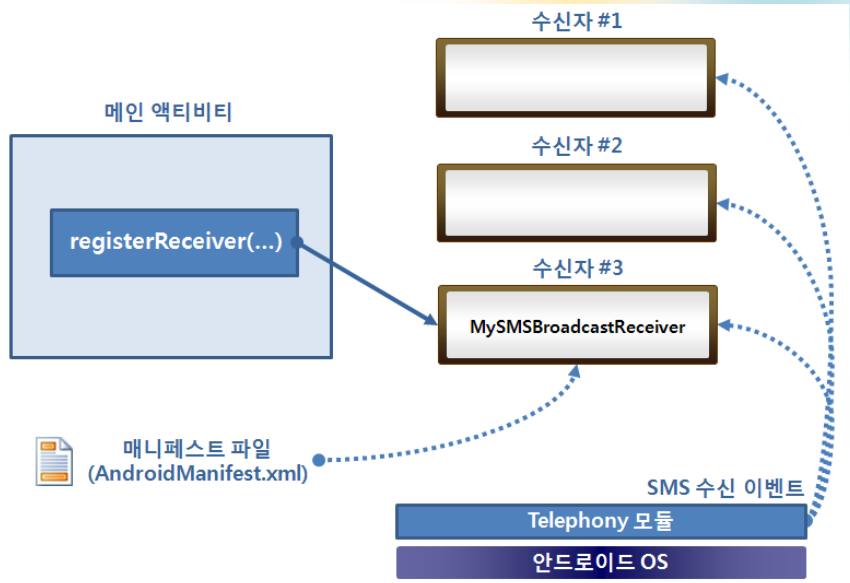
    순서가 같으면 임의로 실행됨.

브로드캐스트 수신자 예제

브로드캐스트 수신자 예제

-브로드캐스트 수신자로 SMS 수신 확인하기

-브로드캐스트 수신자 정의



브로드캐스트 수신자 정의

-일정 시간간격으로 메시지를 보여주는 서비스 클래스 정의

매니페스트에 추가

-새로운 브로드캐스트 수신자를 매니페스트에 추가

브로드캐스트 수신자 클래스 정의

```
public class MySMSBroadcastReceiver extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    Log.i("BroadcastReceiver", "onReceive");
    if (intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
      Log.i("BroadcastReceiver", "SMS Event received!" );

      abortBroadcast();
      Intent myIntent = new Intent(context, MainActivity.class);
      myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
      context.startActivity(myIntent);
  }
  }
}
```

1 브로드캐스트 메시지 수신 시 자동 호출됨

2 SMS_RECEIVED 액션인지를 확인

3 브로드캐스팅 취소

4 새로운 액티비티 띄우기

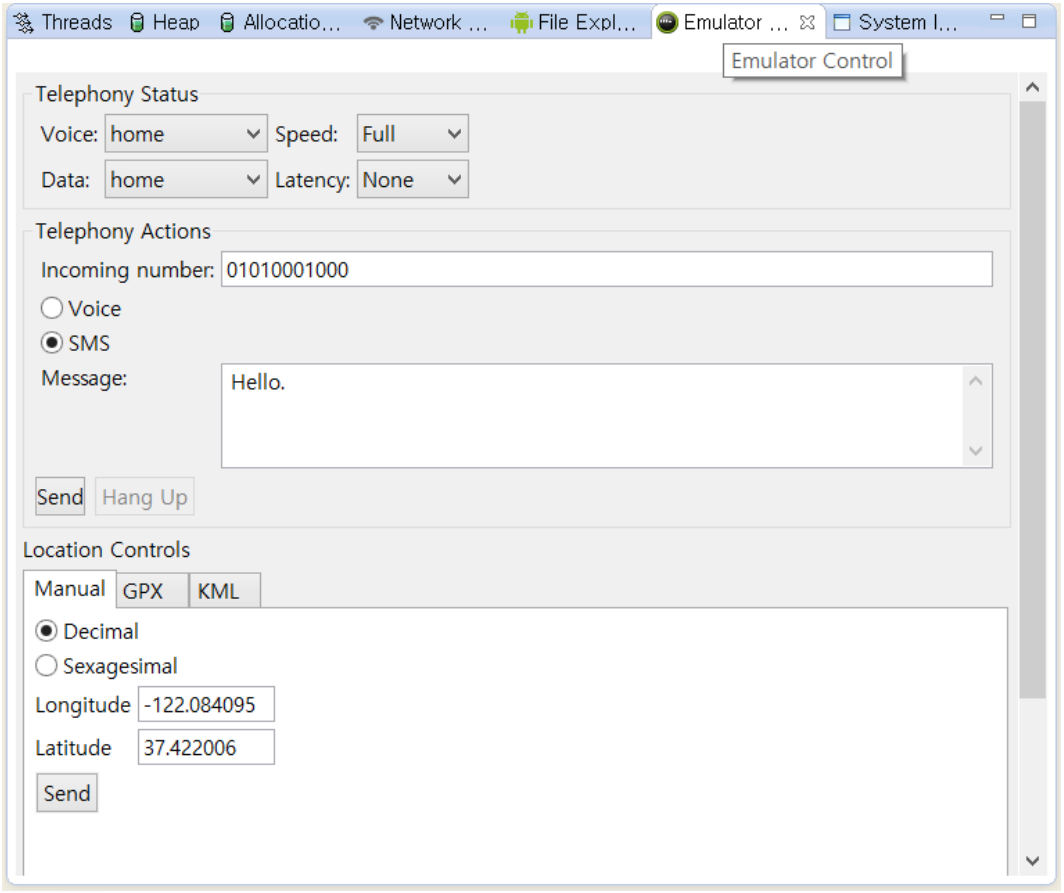
매니페스트에 추가하기

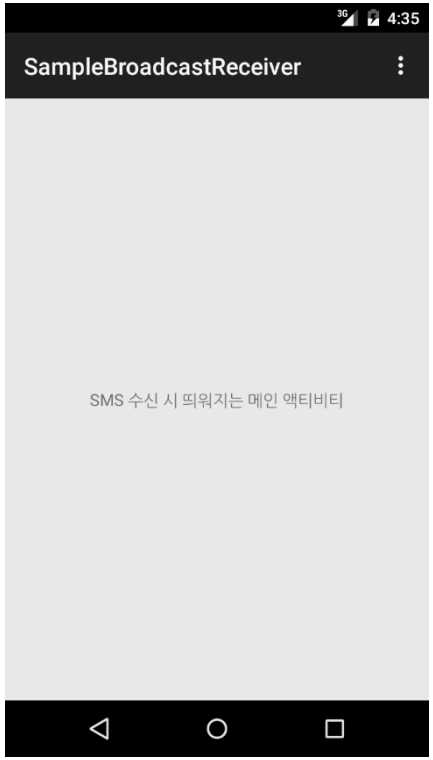
```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
...
  <receiver android:name=".MySMSBroadcastReceiver">
   <intent-filter android:priority="10000">
    <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
   </intent-filter>
  </receiver>
 ...
```

1 SMS 수신 권한 등록

2 수신자 등록

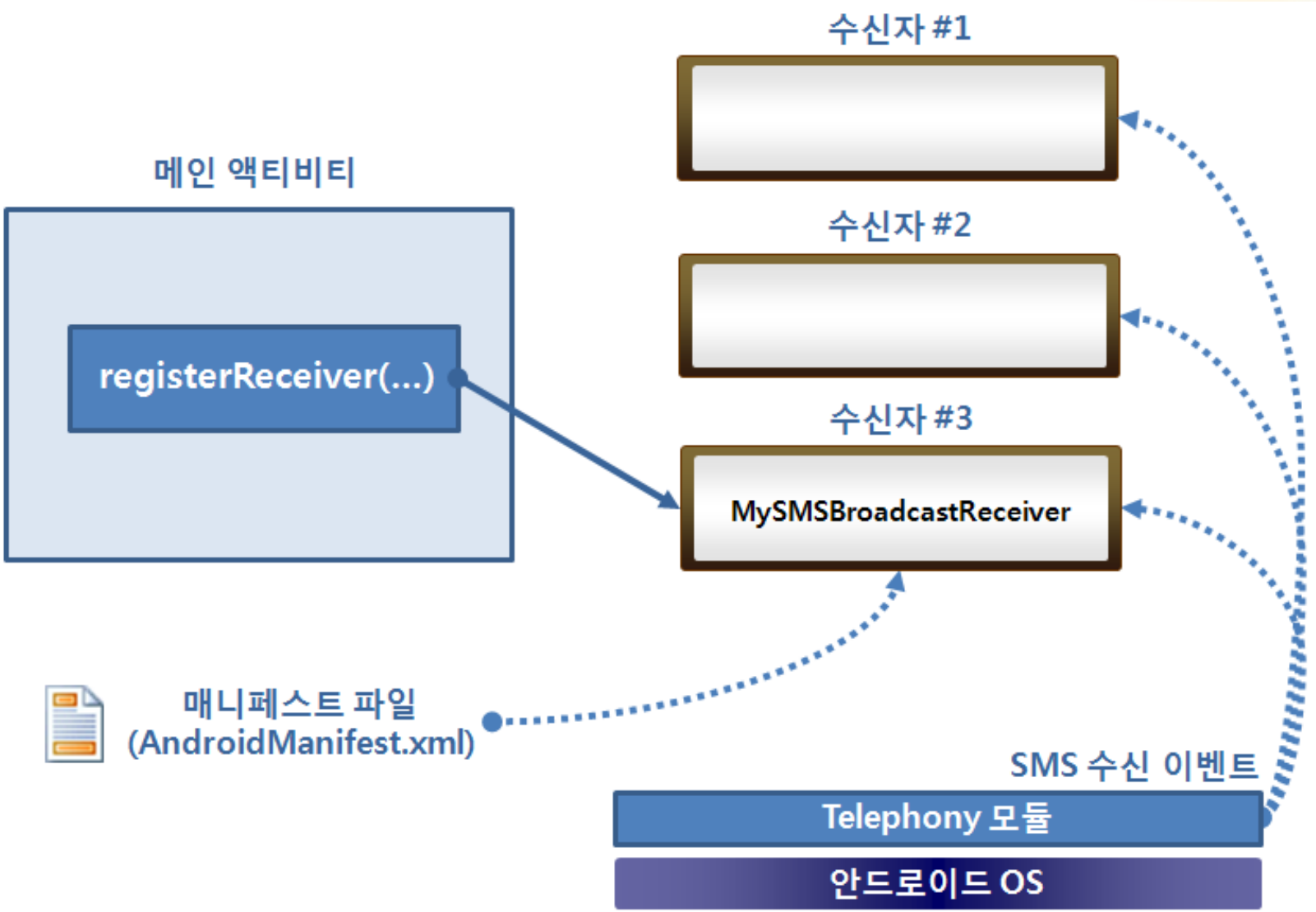
브로드캐스트 수신자 실행 화면





[DDMS에서 에뮬레이터로 SMS를 보내고 메인 액티비티가 뜬 화면]

브로드캐스트 수신자 사용 패턴



7. 앱을 실행했을 때 권한 부여

일반 권한과 위험 권한 (마시멜로 API23부터)

- 위험 권한은 실행 시 권한 부여

대표적인 위험 권한들

LOCATION (위치)
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION

CAMERA
- CAMERA

MICROPHONE
- RECORD_AUDIO

CONTACTS
- READ_CONTACTS
- WRITE_CONTACTS
- GET_ACCOUNTS

PHONE
- READ_PHONE_STATE
- CALL_PHONE
- READ_CALL_LOG
- WRITE_CALL_LOG
- ADD_VOICEMAIL
- USE_SIP
- PROCESS_OUTGOING_CALLS

SMS
- SEND_SMS
- RECEIVE_SMS
- READ_SMS
- RECEIVE_WAP_PUSH
- RECEIVE_MMS

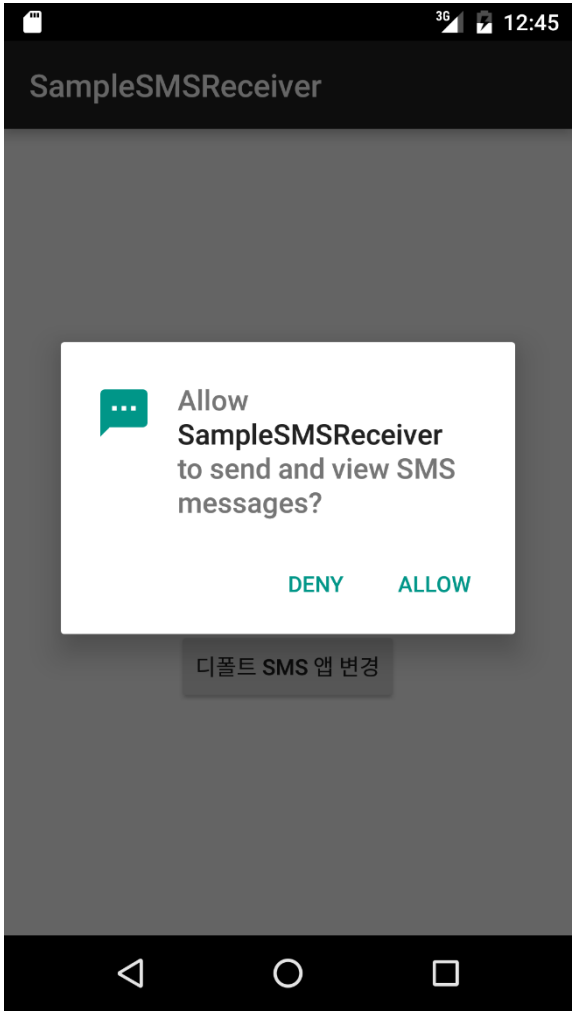
CALENDAR
- READ_CALENDAR
- WRITE_CALENDAR

SENSORS
- BODY_SENSORS

STORAGE
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE

실행 시 권한 부여

- 실행 시 권한 부여를 묻는 대화상자 표시



권한 요청 코드

```
public class MainActivity extends AppCompatActivity {

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

    int permissionCheck = ContextCompat.checkSelfPermission(this,
                        Manifest.permission.RECEIVE_SMS);
    if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
      Toast.makeText(this, "SMS 수신 권한 있음.", Toast.LENGTH_LONG).show();
    } else {
      Toast.makeText(this, "SMS 수신 권한 없음.", Toast.LENGTH_LONG).show();
      if (ActivityCompat.shouldShowRequestPermissionRationale(
           this, Manifest.permission.RECEIVE_SMS)) {
             Toast.makeText(this, "SMS 권한 설명 필요함.",
                              Toast.LENGTH_LONG).show();
      } else {

        ActivityCompat.requestPermissions(this,
                        new String[] {Manifest.permission.RECEIVE_SMS},
                        1);
      }
    }
  }
```

❶

❷

권한 요청 결과 확인 코드

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[],
                                       int[] grantResults) {
  switch (requestCode) {                                          ❶
    case 1: {
      if (grantResults.length > 0 &&
          grantResults[0] == PackageManager.PERMISSION_GRANTED) {
           Toast.makeText(this, "SMS 권한을 사용자가 승인함.",
                          Toast.LENGTH_LONG).show();
      } else {
        Toast.makeText(this, "SMS 권한 거부됨.", Toast.LENGTH_LONG).show();
      }

      return;
    }
  }
}
```

❷

8. 프래그먼트

프래그먼트란?

- **프래그먼트 (Fragment)**
  - 화면의 일정 영역을 독립적으로 처리하기 위해 만들어진 특별한 화면 구성 요소
  - 태블릿의 대화면에서 화면 분할이 필요하게 되면서 만들어짐

- **프래그먼트의 기본 목적**
  - 하나의 화면이 XML 레이아웃과 자바 소스로 구성된다는 점에 착안하여 하나의 프래그먼트가 XML 레이아웃과 자바 소스로 구성되도록 하고 독립적으로 관리되도록 하기 위함

- **기본 구성 방식**
  - 액티비티가 동작하는 방식을 본떠 만들었음

- **사용되는 개념**
  - 프래그먼트 매니저 : 프래그먼트를 관리하는 객체
  - 트랜젝션 : 프래그먼트의 처리를 위해 만든 단위

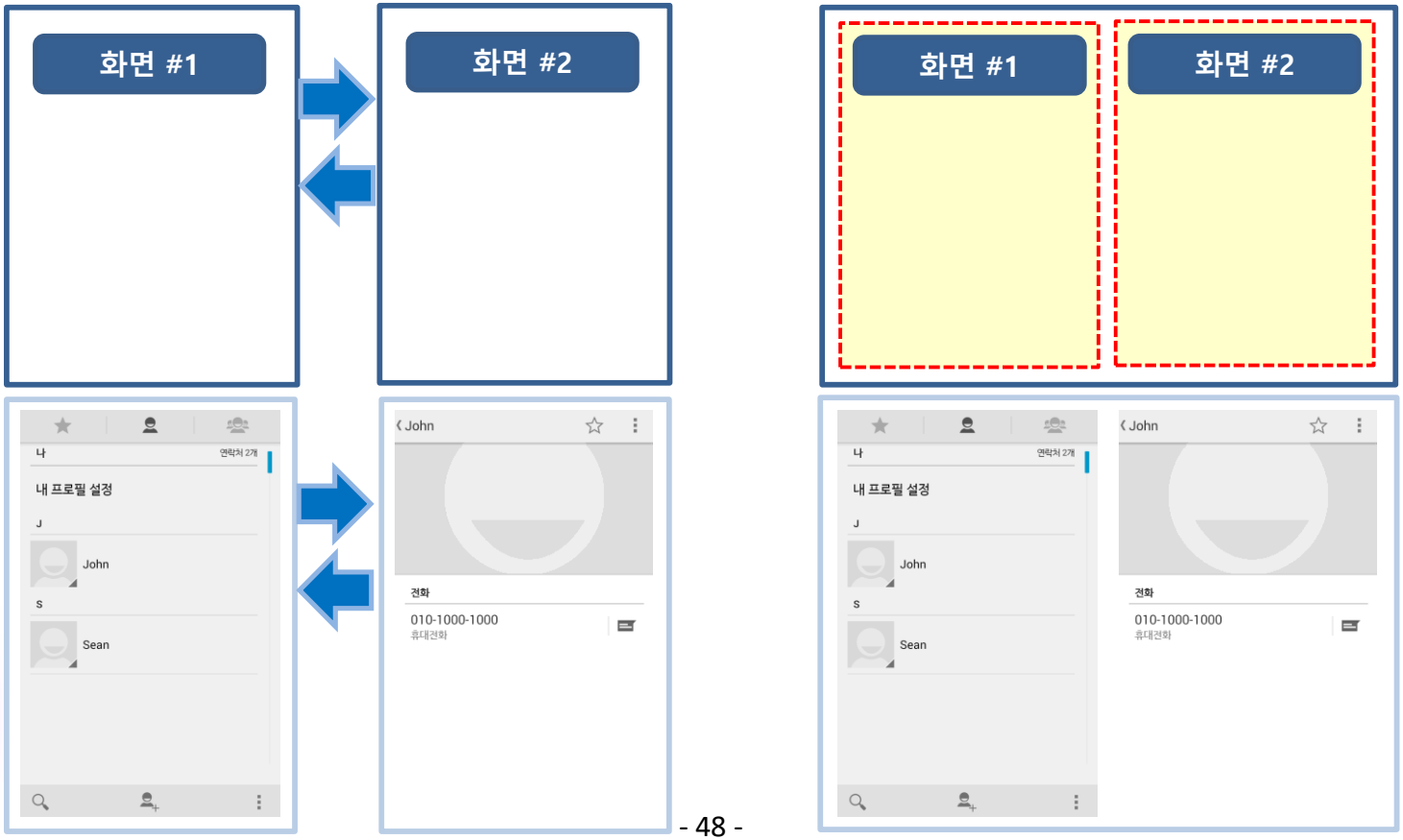
화면을 전환하는 경우와 화면을 분할하는 경우

- **화면 전환과 화면 분할**

  (1) 두 개의 화면을 액티비티로 만들고 액티비티 간 전환

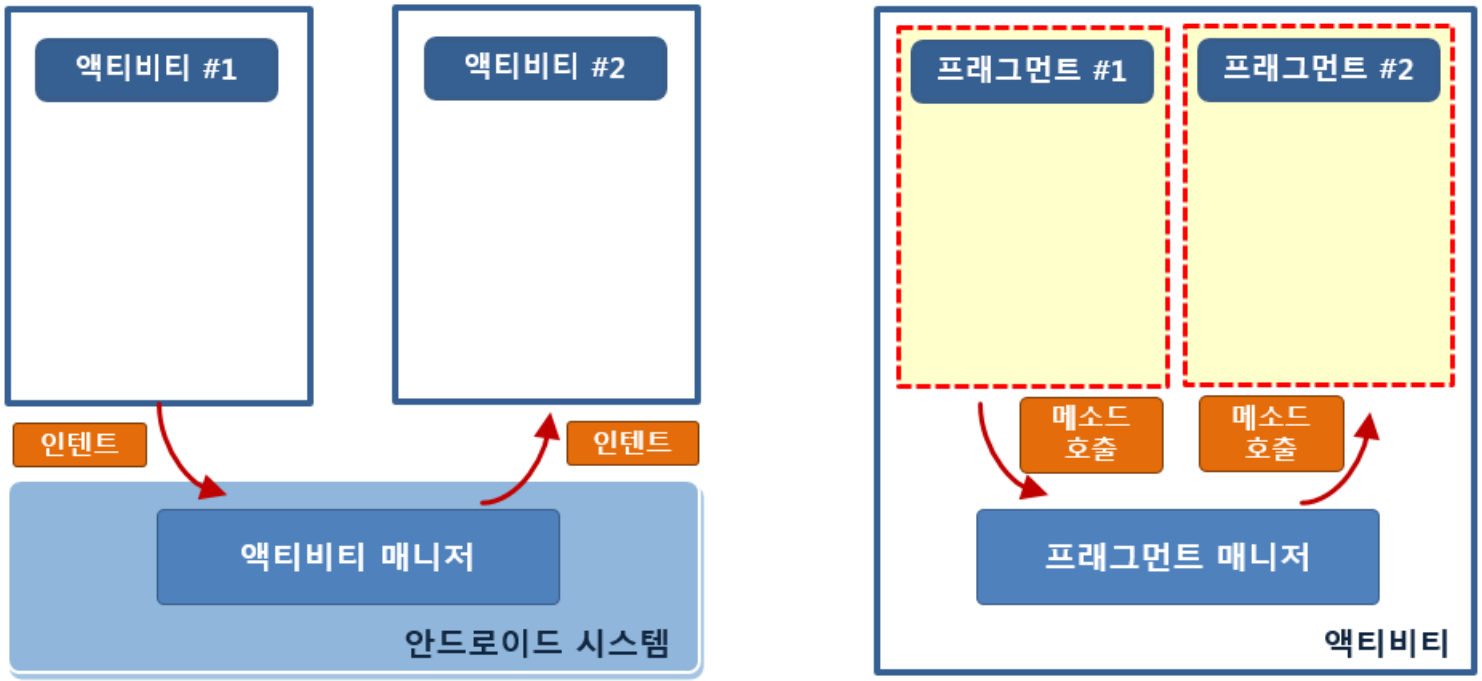
  (2) 하나의 액티비티 위에 프래그먼트를 두고 프래그먼트 간 전환

  ✓ 프래그먼트로 만들어 두면 스마트폰에서는 프래그먼트 간 화면 전환, 태블릿에서는 두 개의 프래그먼트를 하나의 액티비티 위에서 동시에 보여주는 화면 분할이 가능함



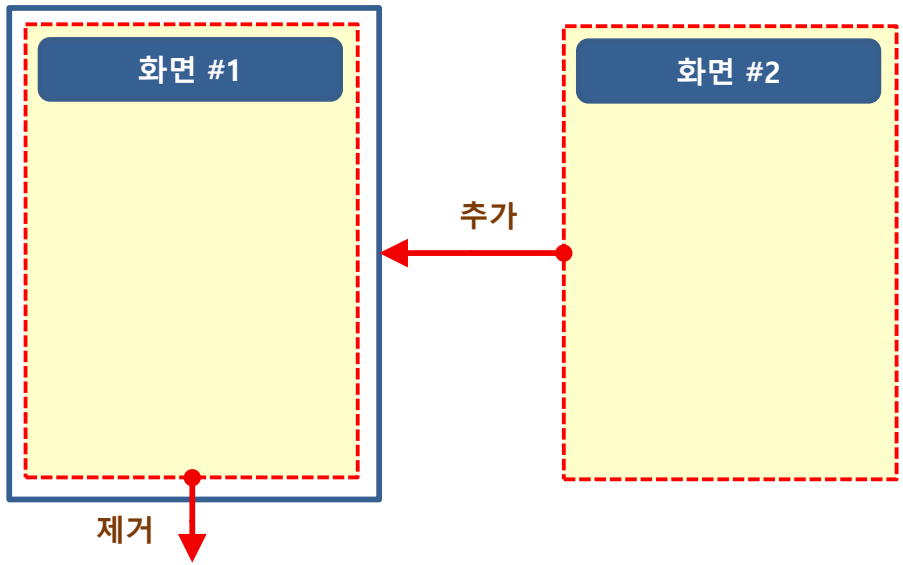
액티비티와 프래그먼트의 동작 방식 비교

- **액티비티를 시스템의 액티비티 매니저에서 관리하듯이 프래그먼트를 액티비티의 프래그먼트 매니저에서 관리함**
  - 프래그먼트 입장에서는 액티비티가 시스템 역할을 하므로 프래그먼트는 항상 액티비티 위에 올라가 있어야 함

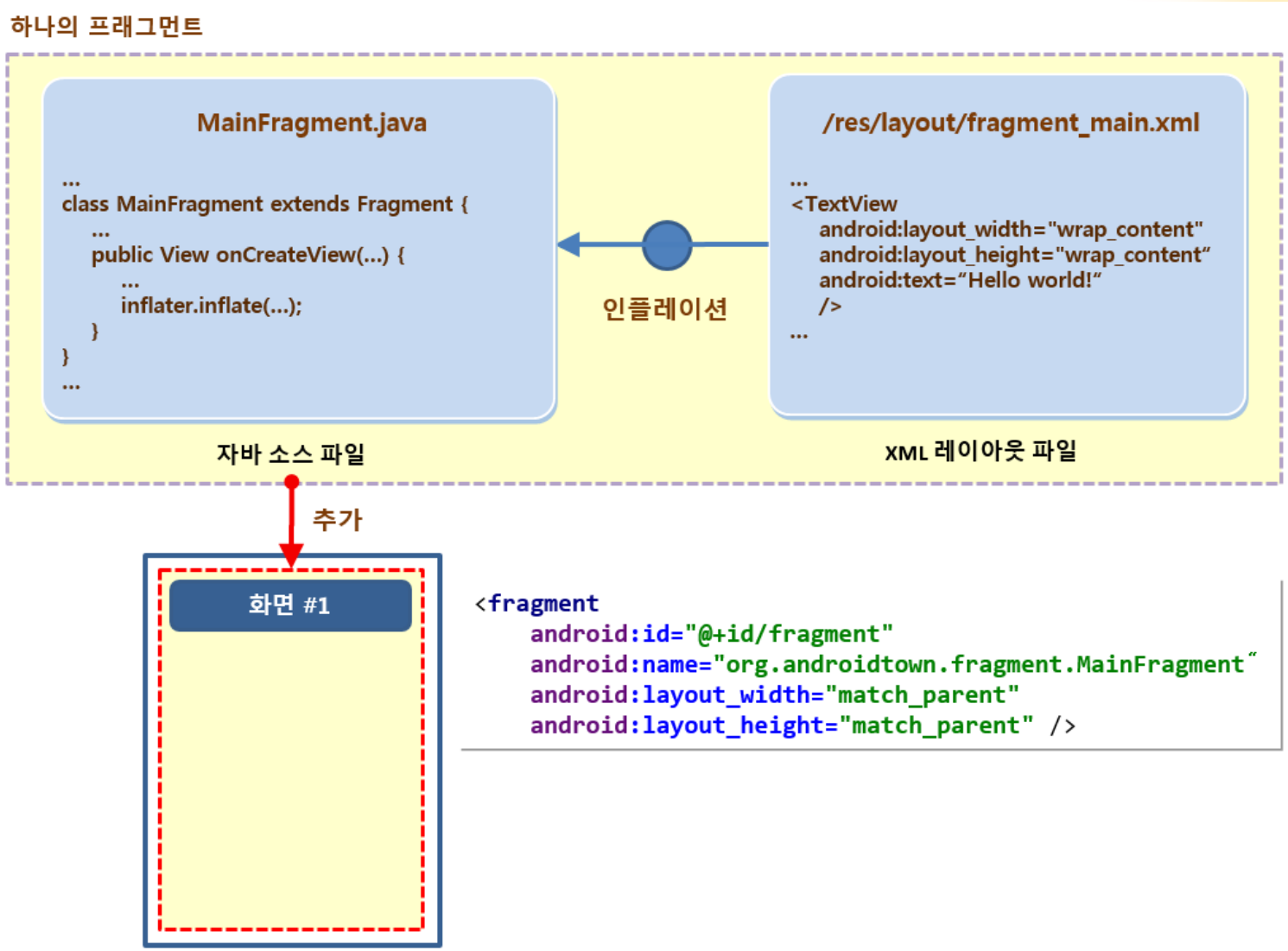


하나의 액티비티 안에서 프래그먼트 전환

- **프래그먼트 전환**
  - 프래그먼트 매니저와 트랜젝션을 이용해 추가(add)나 교체(replace) 가능
  - 싱글 프래그먼트라고 부르며 액티비티 전환 없이 화면 전체가 전환되는 효과를 낼 수 있음



프래그먼트를 위한 레이아웃 인플레이션




```
<fragment
    android:id="@+id/fragment"
    android:name="org.androidtown.fragment.MainFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Fragment 클래스

public final Activity getActivity ()

이 프래그먼트를 포함하는 액티비티를 리턴함.

public final FragmentManager getFragmentManager ()

이 프래그먼트를 포함하는 액티비티에서 프래그먼트 객체들과 의사소통하는 프래그먼트 매니저를 리턴함.

public final Fragment getParentFragment ()

이 프래그먼트를 포함하는 부모가 프래그먼트일 경우 리턴함. 액티비티이면 null을 리턴함.

public final int getId ()

이 프래그먼트의 ID를 리턴함.

FragmentManager 클래스

public abstract FragmentTransaction beginTransaction ()

프래그먼트를 변경하기 위한 트랜젝션을 시작함.

public abstract Fragment findFragmentById (int id)

ID를 이용해 프래그먼트 객체를 찾음.

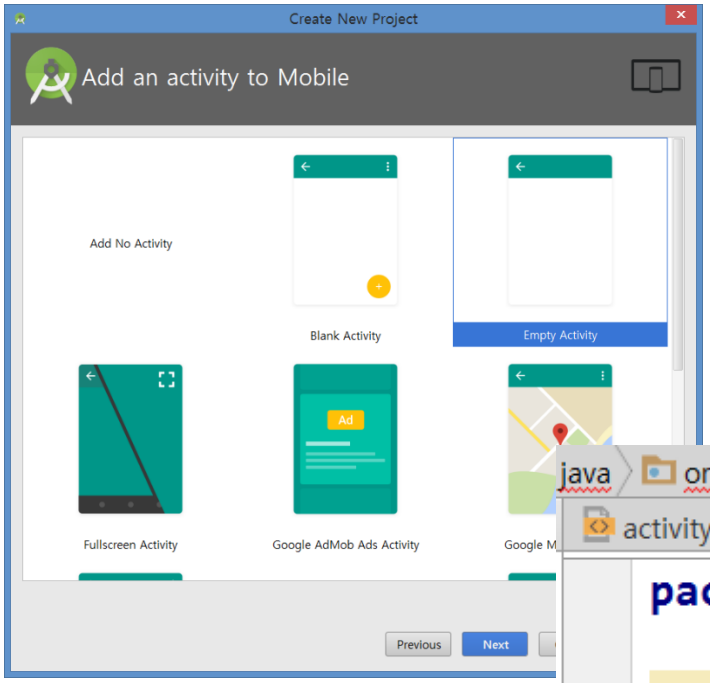
public abstract Fragment findFragmentByTag (String tag)

태그 정보를 이용해 프래그먼트 객체를 찾음.

public abstract boolean executePendingTransactions ()

트랜젝션은 commit() 메소드를 호출하면 실행되지만 비동기(asynchronous) 방식으로 실행되므로 즉시 실행하고 싶다면 이 메소드를 추가로 호출해야 함.

프래그먼트 프로젝트 만들기



java > org > androidtown > fragment > MainFragment

activity_main.xml | MainActivity.java | fragment_main.xml | MainFragment.java

```
package org.androidtown.fragment;

/**
 * Created by user on 2015-12-25
 */
public class MainFragment extends Fragment {

}
```

? android.app.Fragment? (multiple choices...) Alt+Enter

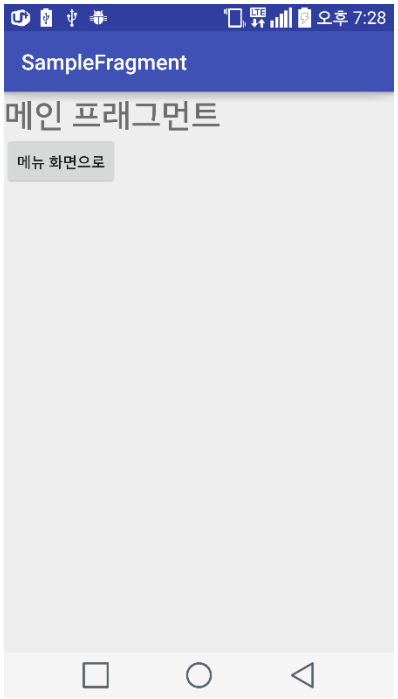
프래그먼트를 만들어 사용하는 과정

(1) 프래그먼트를 위한 XML 레이아웃 만들기

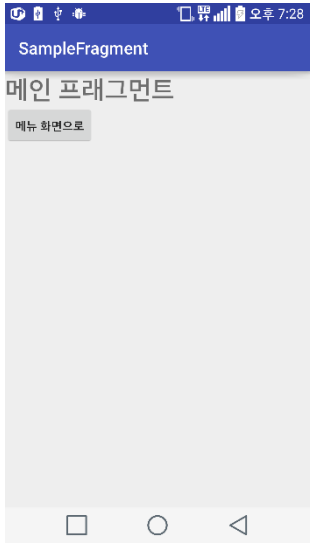
(2) 프래그먼트 클래스 만들기 (클래스 정의)

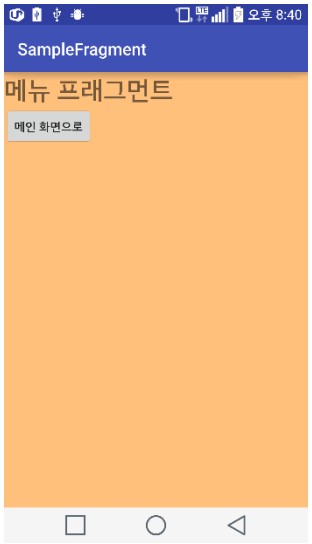
(3) onCreateView() 메소드 안에서 인플레이션하기

(4) 메인 액티비티를 위한 XML 레이아웃에 추가하거나 프래그먼트 매니저를 이용해 코드에서 추가하기



프래그먼트의 교체

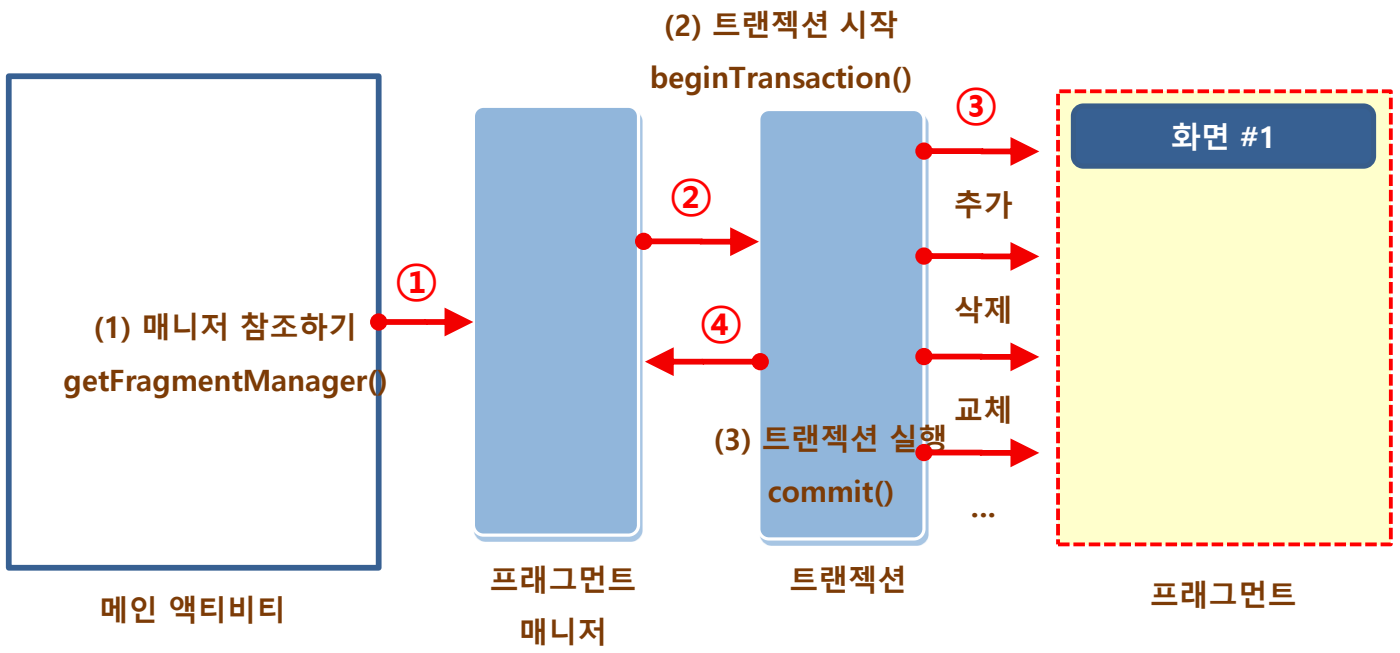




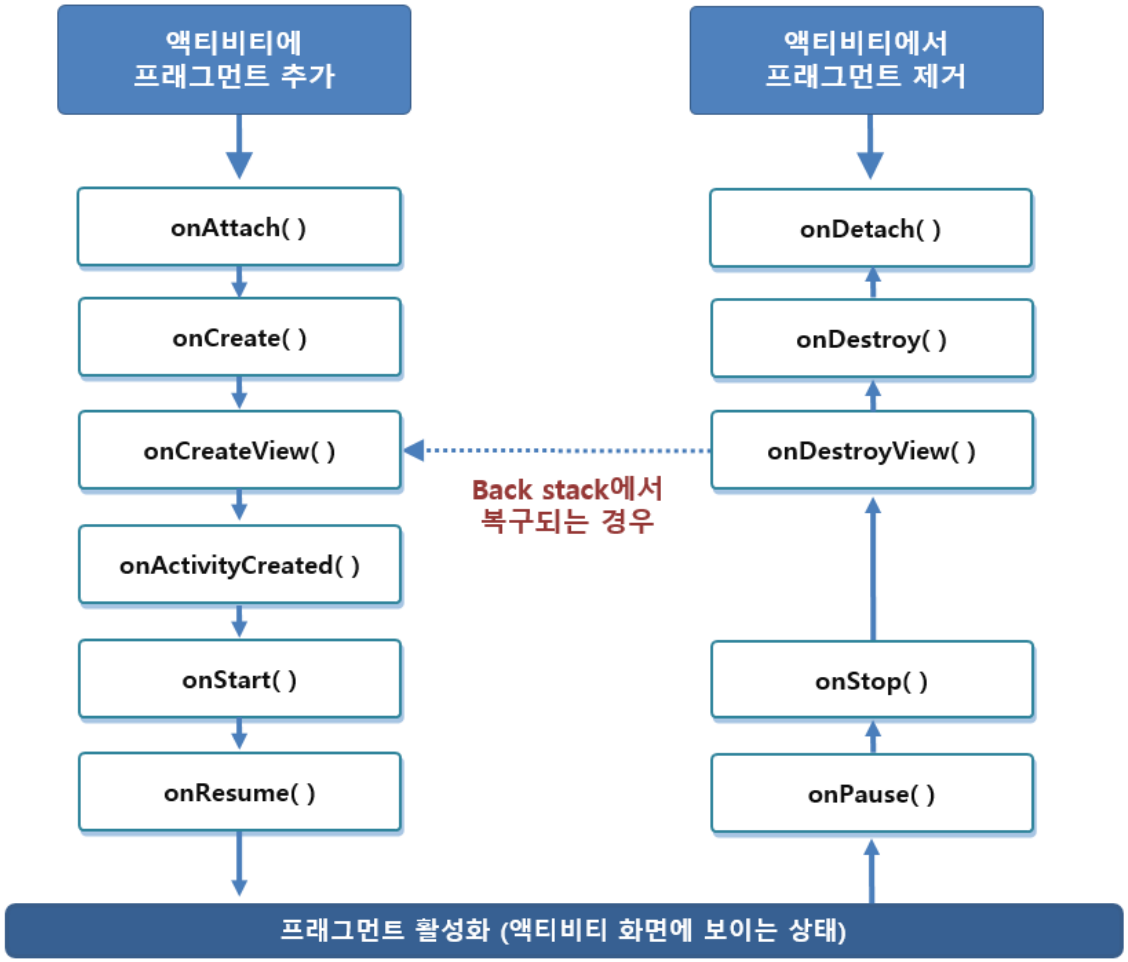

```
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                         @Nullable Bundle savedInstanceState) {
  ViewGroup rootView = (ViewGroup) inflater.inflate(R.layout.fragment_main,
                         container, false);

  Button button = (Button) rootView.findViewById(R.id.button);
  button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      MainActivity activity = (MainActivity) getActivity();
      activity.onFragmentChanged(0);
    }
  });

  return rootView;
}
```

액티비티와 프래그먼트 간 의사소통

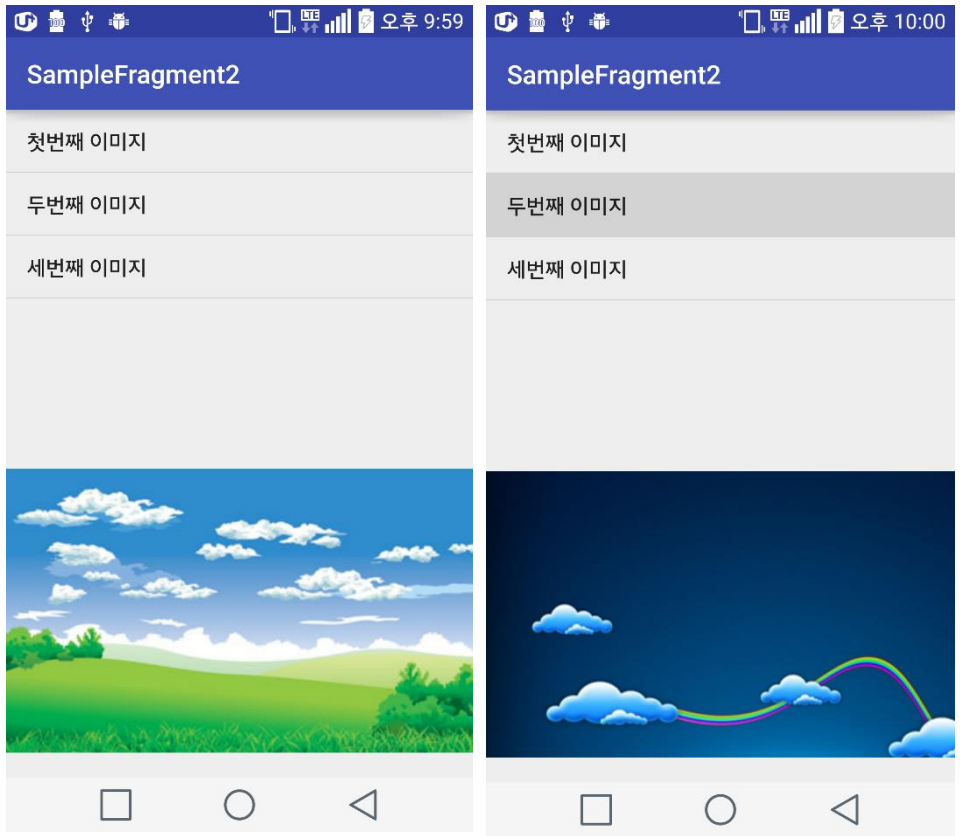
- **프래그먼트 처리 순서**
  - (1) 프래그먼트 매니저 객체 참조
  - (2) 트랜젝션 시작
  - (3) 프래그먼트의 추가, 삭제 또는 교체
  - (4) 트랜젝션 커밋 (commit)



프래그먼트의 수명주기


액티비티에
프래그먼트 추가
onAttach( )
onCreate( )
onCreateView( )
onActivityCreated( )
onStart( )
onResume( )
액티비티에서
프래그먼트 제거
onDetach( )
onDestroy( )
onDestroyView( )
onStop( )
onPause( )
Back stack에서
복구되는 경우
프래그먼트 활성화 (액티비티 화면에 보이는 상태)

한 화면에 두 개의 프래그먼트 넣기




```
<fragment
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:layout_weight="1"
  android:name="org.androidtown.fragment.ListFragment"
  android:id="@+id/listFragment"
  />

<fragment
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:layout_weight="1"
  android:name="org.androidtown.fragment.ViewerFragment"
  android:id="@+id/viewerFragment"
  />
```

액티비티의 코드에서 프래그먼트 전환

```
FragmentManager manager = getSupportFragmentManager();
listFragment = (ListFragment) manager.findFragmentById(R.id.listFragment);
viewerFragment = (ViewerFragment) manager.findFragmentById(R.id.viewerFragment);
```

```
@Override
public void onImageSelected(int position) {
  viewerFragment.setImage(images[position]);
}
```