03. 애플리케이션 구성하기

03-1. 레이아웃 인플레이션 이해하기

: 자동으로 만들어지는 자바 소스 코드(MainActivity.java)를 보면 onCreate() 메소드 안에 있는 코드는 단순히 두 줄뿐이다. super.onCreate() 메소드가 단순히 부모 클래스의 동일한 메소드를 호출한다는 점을 고려하면 **setContentView() 메소드**가 있는 한 줄이 자바 코드의 전부라고 생각할 수도 있습니다. 결국, **어떤 XML 레이아웃 파일과 매칭할 것인지 자바 소스 코드에서 설정하는 부분이 setContentView() 메소드**이란 것을 알 수 있습니다.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); // 이 부분!!
    }
}
```

MainActivity안에는 AppCompatActivity를 상속하는 하나의 클래스가 자동으로 만들어집니다.

- **AppCompatActivity** : 화면에 필요한 기능들을 가지고 있다.
  - **setContentView() 메소드** : XML 레이아웃 파일 이름을 파라미터로 전달하면 XML 레이아웃과 자바 소스 코드가 서로 연결된다.

    ```
    public class MainActivity extends AppCompatActivity {

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

            // 아래와 같이 XML 레이아웃 파일을 불러온다.
            // R.layout.레이아웃 파일 이름
            setContentView(R.layout.activity_main);
        }
    }
    ```

    > **R.layout** : res 폴더 안에, layout 폴더

- **인플레이션(inflation)** : 앱이 실행될 때 XML 레이아웃 파일의 내용이 자바 소스 코드에서 쓰이기 위해 XML이 메모리로 로딩된 후 객체화 되는 과정

- 앱이 실행되는 시점에 XML을 로드하여 메모리에 객체화시킨다. 즉, 실행을 해야만 확인할 수 있다.
- setContentView() 메소드를 실행시키기 전에(인플레이션 하기 전에) 버튼을 불러온다면 오류가 발생한다.

```
public class MainActivity extends AppCompatActivity {

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

        Button button =  (Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(getApplicationContext(), "버튼이 눌렸어요.",
                        Toast.LENGTH_LONG).show();
            }
        });

        // 인플레이션을 하지 않아서 앱을 실행시키면 오류가 발생한다.
        setContentView(R.layout.activity_main);
    }
}
```

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

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

- **LayoutInflater 클래스** : 시스템 서비스로 제공되는 클래스로써, 시스템 서비스는 단말이 시작되면서 항상 실행되는 서비스이다. 단말이 시작되면 단말 안에서 실행되는 기능들을 포함하고 있다.

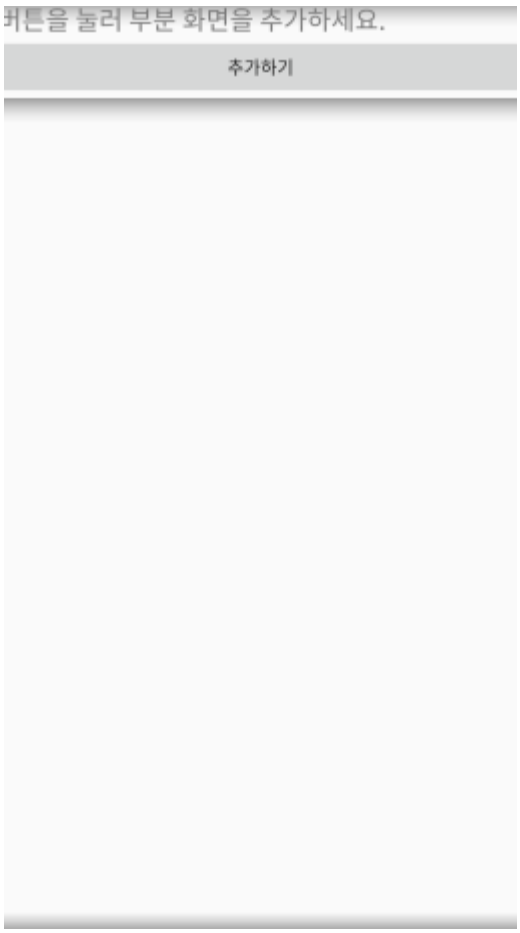
```
// 다음과 같은 코드를 이용하여 LayoutInflater 객체르 참조한 후 사용가능
getSystemService(Context.LAYOUT_INFLATER_SERVICE)
```

예제(내부적 객체화 구현)

참조파일: SampleLayoutInflater>/res/layout/activity_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

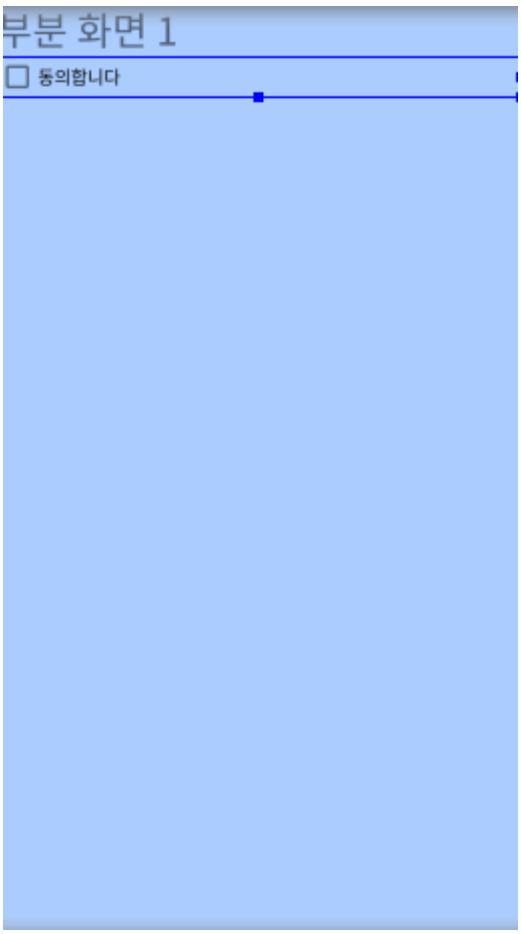
```
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

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

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="버튼을 눌러 부분 화면을 추가하세요."
            android:textSize="20dp"
            />

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

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

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```



참조파일: SampleLayoutInflater> /res/layout/sub1.xml

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

    <LinearLayout
        android:orientation="vertical"
        android:background="#ffaaccff"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView2"
            android:text="부분 화면 1"
            android:textSize="30dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <CheckBox
            android:id="@+id/checkbox"
            android:text="동의합니다"
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content" />

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

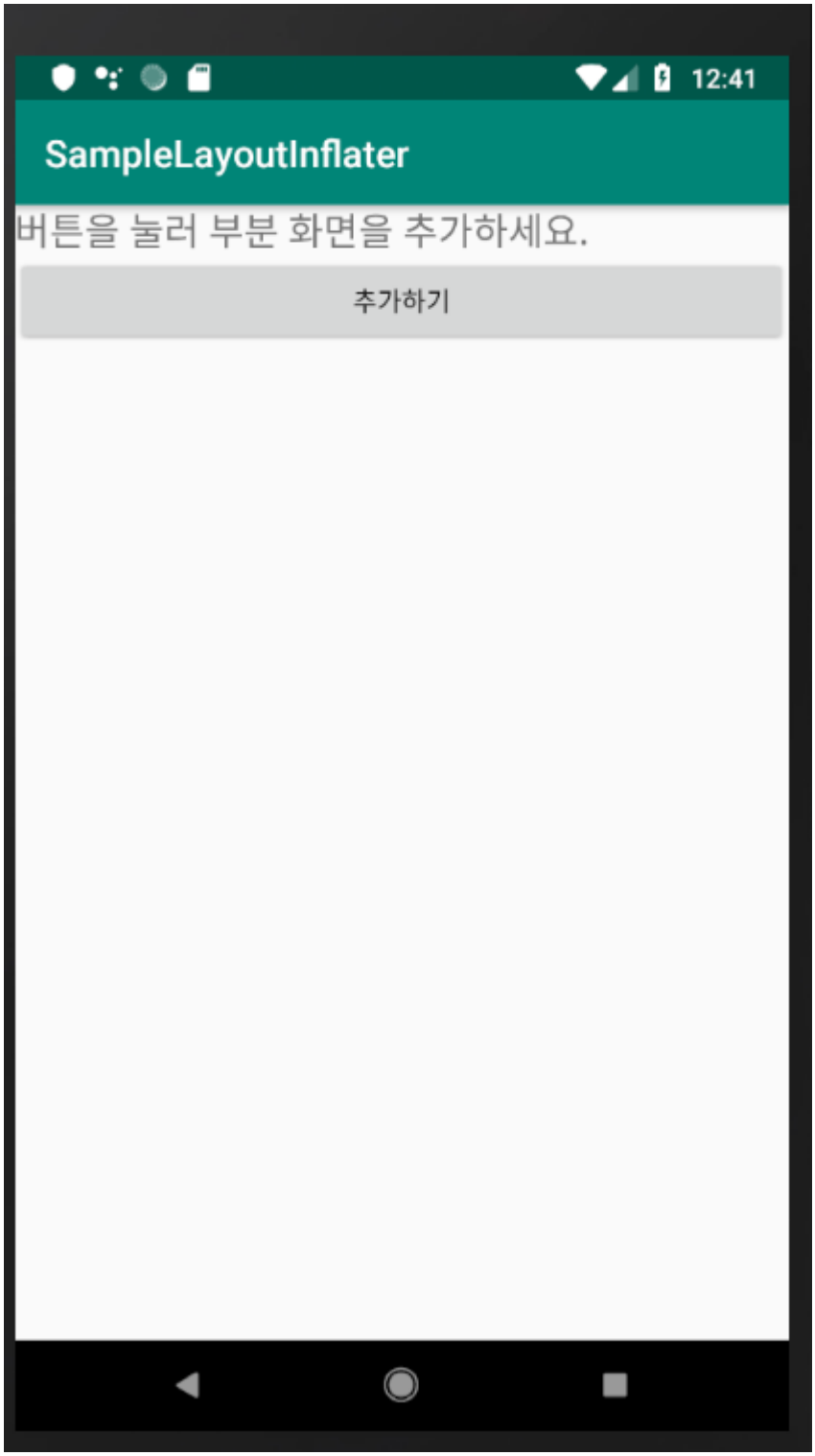


참조파일: SampleLayoutInflater> MenuActivity.java

```
package com.example.samplelayoutinflater;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.LinearLayout;

public class MainActivity extends AppCompatActivity {
    LinearLayout container;     // 리니어 레이아웃 변수를 하나 생성한다.

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

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

        // container 변수를 XML 코드에서 id가 container인 리니어 레이아웃 객체를 참조시킨다.
        container = (LinearLayout) findViewById(R.id.container);

        // button 변수에 XML 코드에서 id가 button2인 버튼을 참조시킨다.
        Button button = (Button) findViewById(R.id.button2);

        // 버튼을 눌렀을 때의 메소드를 익명 객체로 구현
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // getSystemService() 메소드를 사용해서 LayoutInflater 객체를 참조한다.
                LayoutInflater inflater = (LayoutInflater)
                        getSystemService(Context.LAYOUT_INFLATER_SERVICE);

                // <내부적으로 객체화 과정>
                // 참조된 inflater을 inflate() 메소드를 호출해서 sub1과 container 객체를 파라미
터로 전달
                // 이것은 container을 id로 갖는 리니어 레이아웃 객체에 sub1.xml 파일의 레이아웃을
설정하라는 의미이다.
                inflater.inflate(R.layout.sub1, container, true);

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

- **실행 결과**



추가하기 버튼을 눌렀을 때

inflate 관련 메소드들

- **View inflate (int resource, ViewGroup root)**

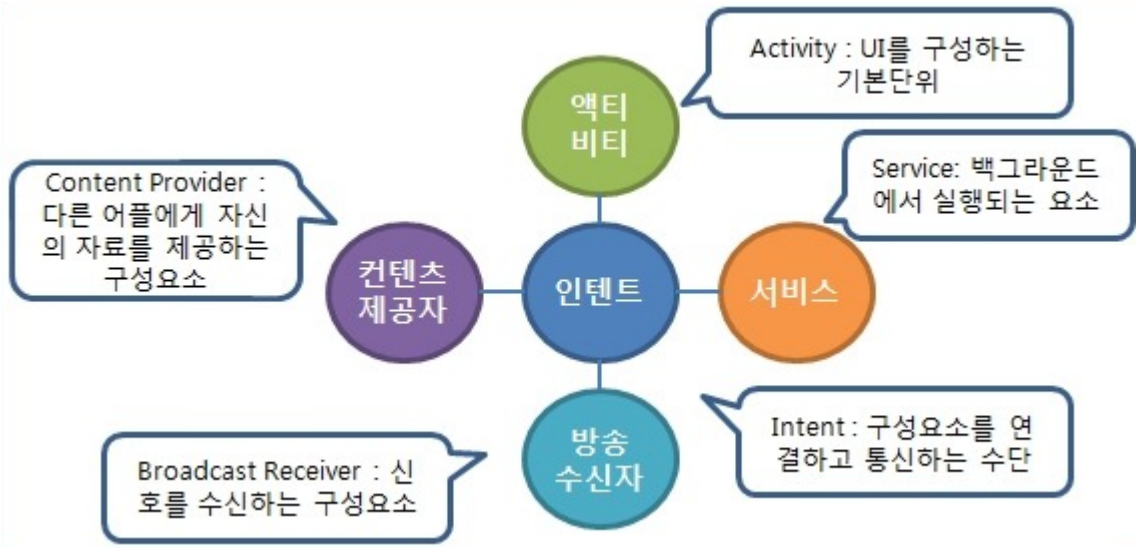
  : 이 메소드의 첫 번째 파라미터로는 XML 레이아웃 리소스를 지정하며, 두 번째 파라미터로는 뷰들을 객체화하여 추가할 대상이 되는 부모 컨테이너를 지정한다.

- **static LayoutInflater.from (Context context)**

  : LayoutInflater를 내부적으로 지원하는 View의 클래스 메소드

- **static View inflate (Context context, int resource, ViewGroup root)** : 다음과 같이 정의된 inflate() 메소드를 이용하면 한 줄로도 객체화 과정을 수행할 수 있다.

03-2. 화면 구성과 화면 간 전환



- 하나의 화면을 하나의 **액티비티**라고 한다.
- **AndroidManifest.xml** : 액티비티, 서비스, 브로드캐스트 수신자, 내용 제공자들의 정보를 가지고 있다. 애플리케이션을 구성하는 구성 요소 네 가지는 새로 만들때마다 항상 그 정보를 매니페스트 파일에 추가해야 한다.

- **startActivity() 메소드**

  : 액티비티를 띄워 화면에 보이도록 만드는 메소드

- **startActivityForResult(Intent intent, int requestCode) 메소드**

  : 이 메소드의 파라미터는 인텐트와 정수로 된 코드 값인데 이 코드 값은 각각의 액티비티를 구분하기 위해 사용된다. 새로 띄웠던 여러 액티비티 중에 어떤 것으로부터 온 응답인지 구분할 필요가 있을때 이 메소드를 사용한다.

예제(액티비티 간의 데이터 전송)

- **AndroidManifest.xml**

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

```
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <!--메뉴 엑티비티를 지우고-->
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!--액티비티를 새로 적는다.-->
        <activity
            android:name=".MenuActivity"
            android:label="메뉴 엑티비티"
            android:theme="@style/Theme.AppCompat.Dialog">
        </activity>
    </application>

</manifest>
```

- **activity_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MenuActivity">

    <!--정가운데 배치-->
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="돌아가기"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
</android.support.constraint.ConstraintLayout>
```

- **MenuActivity.java**

```
package com.example.sampleintent;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MenuActivity extends AppCompatActivity {

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

        Button button = (Button)findViewById(R.id.button);      // 버튼 객체 참조

        // 버튼 익명 객체 구현
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent();                   // 인텐트 객체 생성

                // name의 값을 부가 데이터로 넣기, 키와 데이터 값을 쌍으로 넣어주어야 한다.
                // 다시 확인하고자 할 경우에는 키를 이용해 데이터 값을 가져올 수 있다.
                intent.putExtra("name", "mike");

                // 응답 보내기, 새로 띄운 액티비티에서 이전 액티비티로 인텐트를 전달하고 싶을 때
사용되는 메소드
                // 형식, setResult(응답 코드, 인텐트)
                setResult(RESULT_OK, intent);

                finish();   // 현재 액티비티 없애기
            }
        });
    }
}
```

- **MainActivity.java**

```
package com.example.sampleintent;

import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {
    // 다른 액티비티를 띄우기 위한 요청코드 정의
    public static final int REQUEST_CODE_MENU = 101;

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

    @Override
    // 이 메소드의 첫 번째 파라미터는 액티비티 요청 코드이다. 즉, 어떤 액티비티로부터 응답을 받은 것인지 구분할 때 사용된다.
    // 두 번째 파라미터는 응답을 보내 온 액티비티로부터 전달된 응답 코드이다. 보통 Activity.RESULT_OK 상수로 정상 처리 됨을 알려준다.
    // 또한 임의로 만든 코드를 전달할 수도 있다.
    // 세 번째 파라미터는 새로 띄웠던 메뉴 액티비티로 부터 전달 받은 인텐트이다. 이 인텐트 객체는 메뉴 액티비티로 부터
    // 메인 액티비티로 데이터를 전달할 목적으로 사용된다.
    // 이 메소드는 새로 띄웠던 메뉴 액티비티가 응답을 보내오면 그 응답을 처리하는 역할을 한다.
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_CODE_MENU) {
            Toast.makeText(getApplicationContext(), "onActivityResult 메소드 호출됨. 요청 코드 : " + requestCode + ", 결과 코드 : "
                    + resultCode, Toast.LENGTH_LONG).show();

            if (requestCode == RESULT_OK) {
                String name = data.getExtras().getString("name");
                Toast.makeText(getApplicationContext(), "응답으로 전달된 name : " + name, Toast.LENGTH_LONG).show();
            }
        }
    }

    public void onButton1Clicked(View v) {
        // 또 다른 액티비티를 띄우기 위한 인텐트 객체 생성
        Intent intent = new Intent(getApplicationContext(), MenuActivity.class);

        // 액티비티 띄우기, 액티비티를 띄울 목적으로 사용될 수도 있고, 액티비티 간에 데이터를 전달하는 데에도 사용될 수 있다.
        startActivityForResult(intent, REQUEST_CODE_MENU);
    }
}
```

- **실행결과**

메뉴화면 띄우기 클릭시

돌아가기 클릭시

액티비티 추가와 요청 그리고 응답 과정

1. **새로운 액티비티 만들기**

   : 새로운 액티비티를 추가하면 XML 레이아웃 파일 하나와 자바 소스 파일 하나가 만들어지고 매니페스트 파일에 액티비티 태그가 추가된다.

2. **새로운 액티비티의 XML 레이아웃 정의하기**

   : 새로 만들어진 XML 레이아웃을 수정하여 새로운 액티비티의 화면이 어떻게 배치될지를 작성한다.

3. **메인 액티비티에서 새로운 액티비티 띄우기**

   : 메인 액티비티의 버튼을 클릭하면 startActivityForResult() 메소드로 새로운 액티비티를 띄운다.

4. **새로운 액티비티에서 응답 보내기**

   : 새로운 액티비티가 보이고 그 안에 들어 있는 버튼을 클릭하면 setResult() 메소드로 응답을 보낸다.

5. **응답 처리하기**

   : 메인 엑티비티에서 onActivityResult() 메소드를 재정의하여 새로 띄웠던 액티비티에서 보내오는 응답을 처리한다.

03-3. 인텐트 살펴보기

- **Intent** : 이전 과정에서 사용된 인텐트는 다른 액티비티를 띄우거나 기능을 동작시키기 위한 수단으로 사용되었다. 즉, **무언가 작업을 수행하기 위해 사용되는 일종의 명령 또는 데이터 전달 수단으로 사용된다.**

  인텐트를 만든 후 startActivity() 또는 startActivityForResult() 메소드를 호출하면서 전달하면 이 인텐트는 시스템으로 전달된다.

  이렇게 인텐트는 **애플리케이션 구성 요소 간에 작업 수행을 위한 정보를 전달하는 역할을 한다.**

인텐트의 역할과 사용 방식

- **다른 애플리케이션 구성 요소에 인텐트를 전달할 수 있는 대표적인 메소드**
  - startActivity(), startActivityForResult()

    : 화면을 띄울 때 사용된다.
  - startService(), bindService()

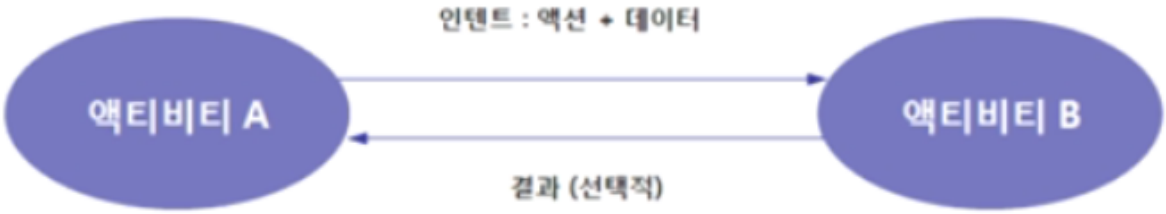
    : 서비스를 시작할 때 사용된다.
  - broadcastIntent()

    : 브로드캐스팅을 수행할 때 사용된다.

- **인텐트의 기본 구성 요소**
  - 액션(Action) : 수행할 기능

    예시

    - 예를 들어 인텐트 객체를 만들 때 ACTION_VIEW와 함께 웹페이지 주소를 전달하면 단말 안에 설치되어 있던 웹브라우저의 화면이 뜨면서 해당 웹페이지를 보여준다.
  - 데이터(Data) : 액션이 수행될 대상 데이터



[액티비티 간의 인텐트 전달]

액션과 데이터를 사용하는 대표적인 예

속성	설 명
ACTION_DIAL tel:01011112222	주어진 전화번호를 이용해 전화걸기 화면을 보여줌.
ACTION_VIEW tel:01011112222	주어진 전화번호를 이용해 전화걸기 화면을 보여줌. URI 값의 유형에 따라 VIEW 액션이 다른 기능을 수행
ACTION_EDIT content://contacts/people/2	전화번호부 데이터베이스에 있는 정보 중에서 ID 값이 2인 정보를 편집하기 위한 화면을 보여줌
ACTION_VIEW content://contacts/people/	전화번호부 데이터베이스의 내용을 보여줌

- **명시적 인텐트(Explicit Intent)** : 인텐트에 클래스 객체나 컴포넌트 이름을 지정하여 호출할 대상을 확실히 알 수 있는 경우

- **암시적 인텐트(Implicit Intent)** : 액션과 데이터를 지정하긴 했지만 호출할 대상이 달라질 수 있는 경우
  - **범주(Category)**

    : 액션이 실행되는 데 필요한 추가적인 정보를 제공한다.
  - **타입(Type)**

    : 인텐트에 들어가는 데이터의 MIME 타입을 명시적으로 지정한다.
  - **컴포넌트(Component)**

    : 인텐트에 사용될 컴포넌트 클래스 이름을 명시적으로 지정한다.
  - **부가 데이터(Extra Data)**

    : 인텐트는 추가적인 정보를 넣을 수 있도록 번들(Bundle) 객체를 담고 있다. 이 객체를 통해 인텐트 안에 더 많은 정보를 넣어 다른 애플리케이션 구성 요소에 전달할 수 있다.

예제(인텐트를 사용해서 전화걸기)

- **activity_main.xml**

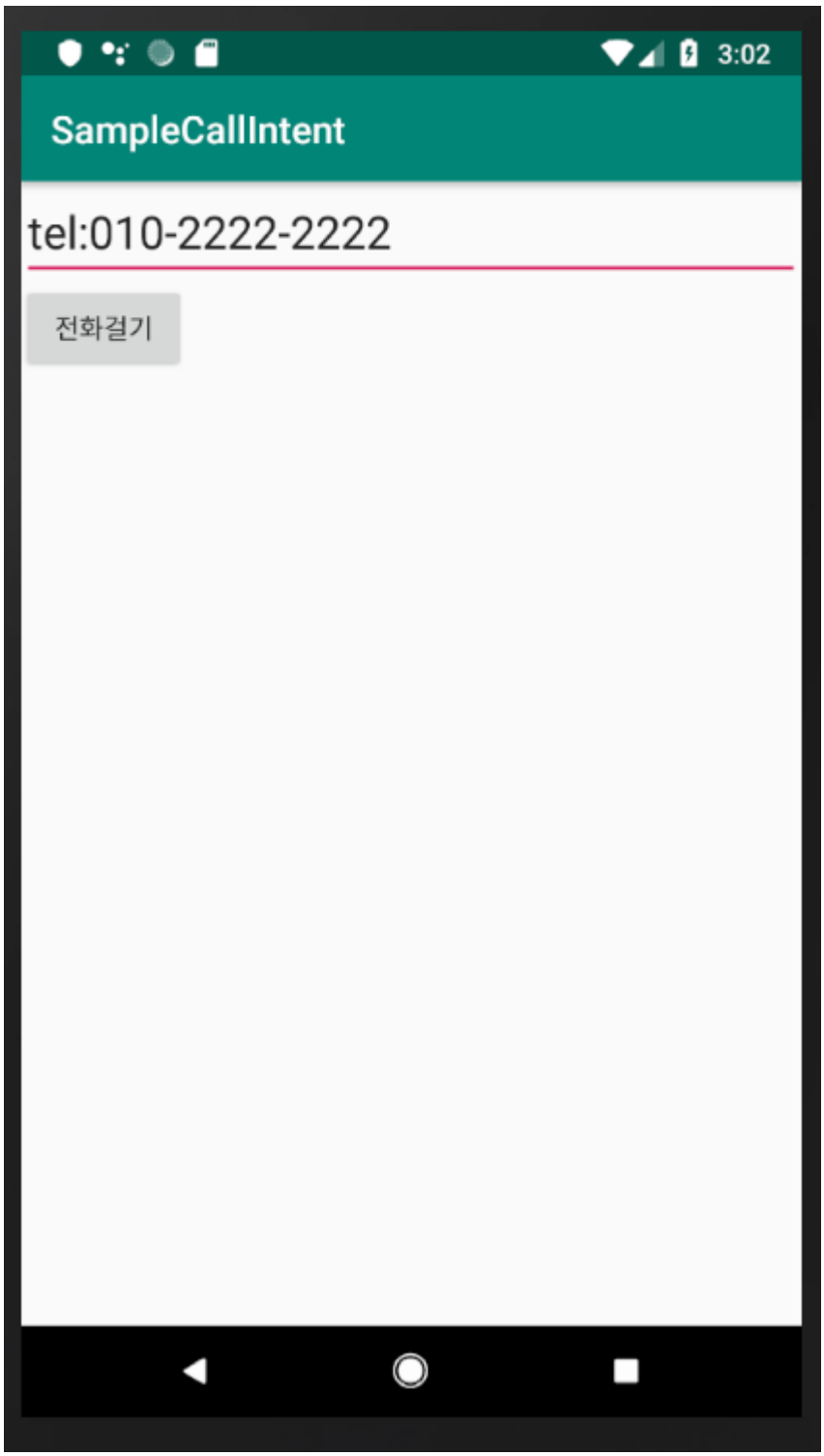
```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
   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

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

        <!--전화번호를 입력할 입력상자 정의-->
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/editText"
            android:text="tel:010-1000-1000"
            android:textSize="24dp"
            />

        <!-- 전화걸기 버튼 정의 -->
        <Button
            android:id="@+id/button"
            android:text="전화걸기"
            android:onClick="onButton1Clicked"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

- **MainActivity.java**

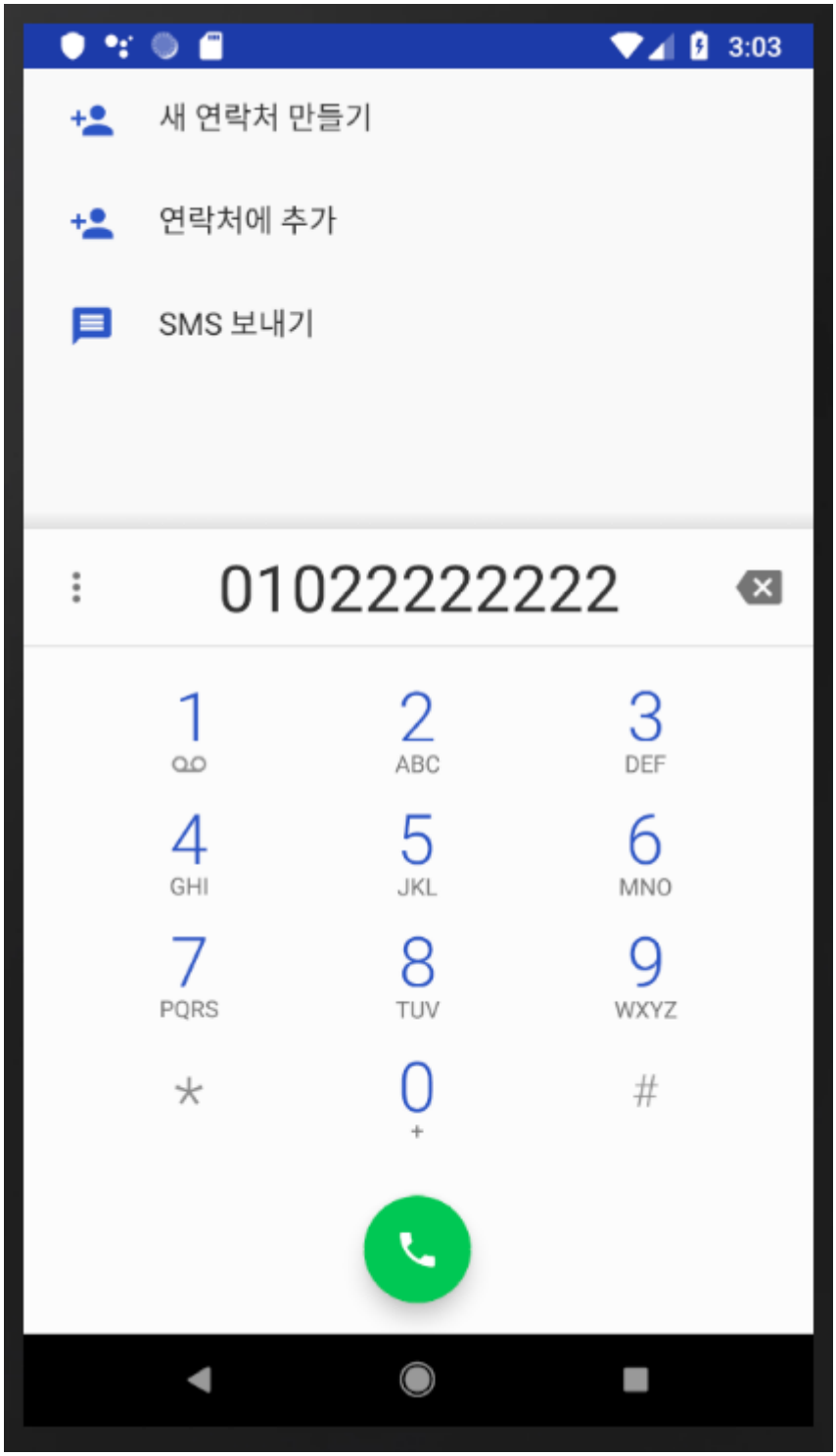
```java
package com.example.samplecallintent;

import android.content.Intent;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    EditText editText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        editText = (EditText)findViewById(R.id.editText);   // 뷰 객체 참조

        Button button = (Button)findViewById(R.id.button);  // 버튼 객체 참조

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 입력 상자에 입력된 전화번호 확인
                String data = editText.getText().toString();

                // 전화걸기 화면을 보여줄 인텐트 객체 생성
                Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse(data));

                // 액티비티 띄우기
                startActivity(intent);
            }
        });
    }
}
```

- **실행 결과**

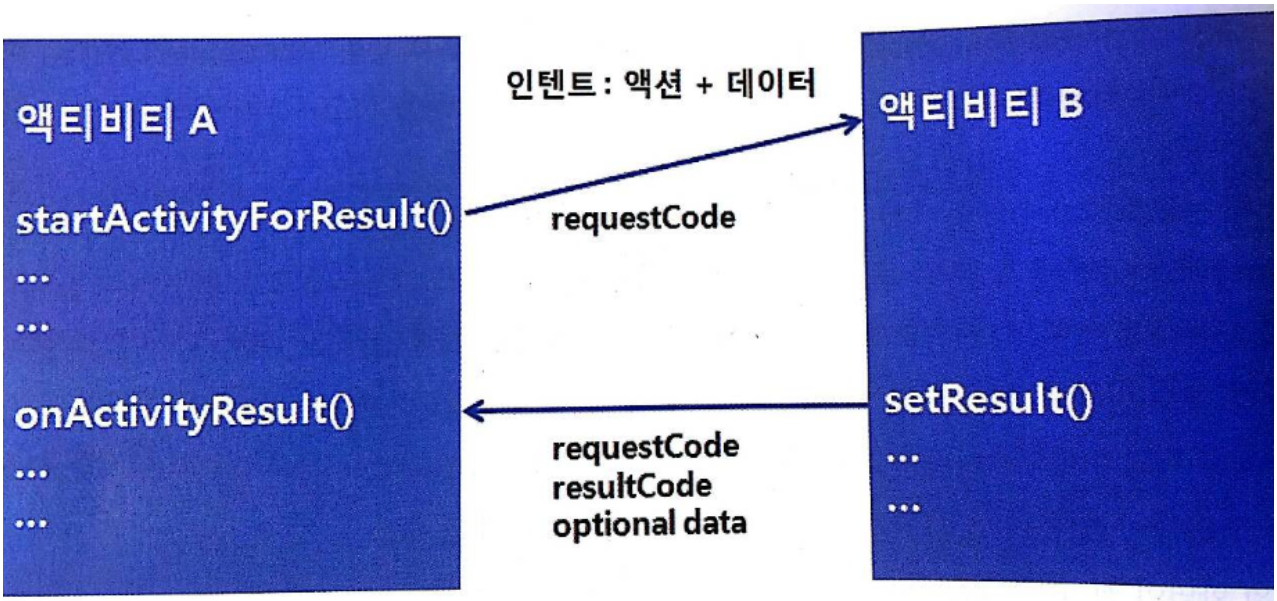


전화걸기 클릭시



PDF 파일 보여주기

- **startActivity()** : 새로 띄우는 다른 액티비티로부터 받는 응답을 처리할 필요가 없을 때 간단하게 사용
- **startActivityForResult()** : 대부분의 실제 앱에서는 응답 처리가 필요한 경우가 많으므로 이 메소드를 사용한다.

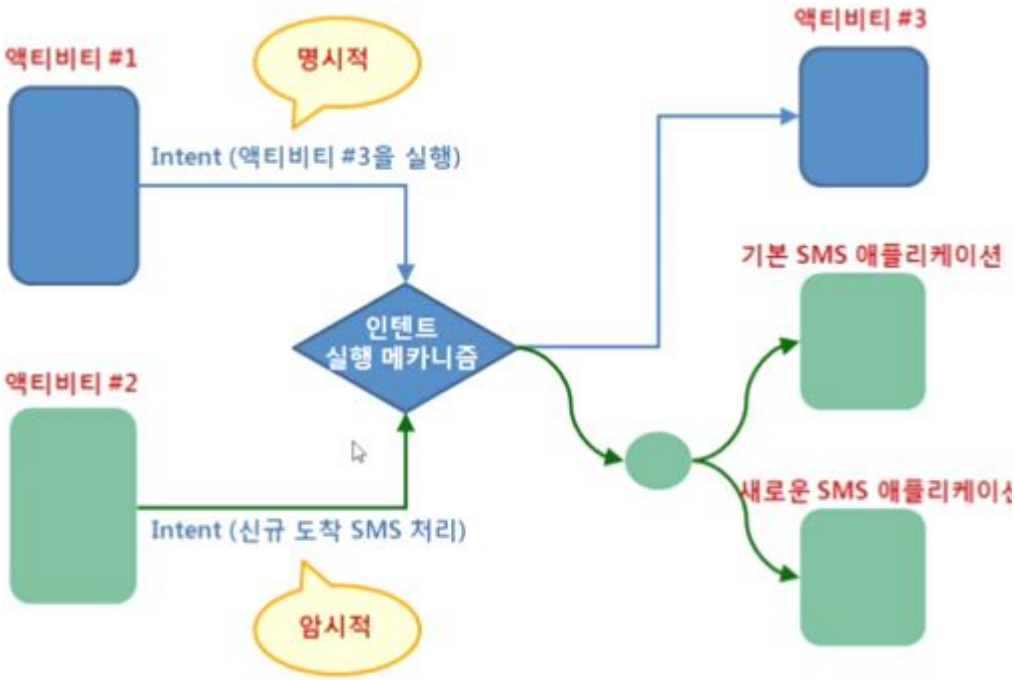


▲ startActivityForResult() 메소드를 이용한 액티비티 간의 전환

- **결과 값 코드(resultCode)** : 성공이나 실패를 의미하는 상수, onActivityResult() 메소드에서 값을 받을 수 있다.
  - Activity.RESULT_OK, 성공
  - Activity.RESULT_CANCELED, 실패

- **암시적 인텐트 역할** : 인텐트 안의 데이터를 해석한 후 단말에 설치된 앱 중에서 이 데이터를 처리하기 적절한 것을 찾아야 한다. 즉, 중개(Mediation) 역할을 한다.

- **인텐트 필터(Intent Filter)** : 시스템이 요청하는 인텐트의 정보를 받아 처리할 애플리케이션 구성 요소를 찾기 위해 필요한 정보



: 명시적 인텐트를 이용하면 명시한 액티비티를 실행시키고 암시적 인텐트를 이용하면 인텐트 필터를 통해 암시적 인탠트와 동일한 액션 정보를 가진 앱을 실행시킨다.

예제(암시적 인텐트로 PDF 문서 보여주기)

- **activity_main.xml**

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

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

        <!-- 파일명 입력상자 정의 -->
        <EditText
            android:id="@+id/editText"
            android:textSize="24dp"
            android:hint="PDF 파일명을 입력하세요."
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <!-- 열기 버튼 정의 -->
        <Button
            android:id="@+id/button"
            android:text="열기"
            android:onClick="onButton1Clicked"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

- **MainActivity.java**

```
package com.example.samplepdfview;

import android.content.ActivityNotFoundException;
import android.content.Intent;
```

```
import android.net.Uri;
import android.os.Environment;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import java.io.File;

public class MainActivity extends AppCompatActivity {

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

    public void onButton1Clicked(View v) {
        EditText editText = (EditText)findViewById(R.id.editText);

        // 입력상자에 입력된 파일명 확인
        String filename = editText.getText().toString();

        if (filename.length() > 0 ) {
            // openPDF() 메소드 호출
            openPDF(filename.trim());
        } else {
            Toast.makeText(getApplicationContext(), "PDF 파일명을 입력하세요.",
                    Toast.LENGTH_LONG).show();
        }
    }

    // PDF 파일 열기 기능을 정의한 메소드
    public void openPDF(String filename) {
        // SD 카드 폴더의 패스 추가. SD 카드에 저장되어 있는 PDF 파일을 지정하기 위해서
        String sdcardFolder =
Environment.getExternalStorageDirectory().getAbsolutePath();
        String filepath = sdcardFolder + File.separator + filename;
        File file = new File(filepath);

        if (file.exists()) {
            // Uri 객체로 생성
            Uri uri = Uri.fromFile(file);

            // ACTION_VIEW 액션을 가지는 인텐트 객체 생성
            Intent intent = new Intent(Intent.ACTION_VIEW);

            // Uri 객체와 MIME 타입 지정
            intent.setDataAndType(uri, "application/pdf");

            try {
                // 액티비티 띄우기
```

```
                startActivity(intent);
            } catch (ActivityNotFoundException ex) {
                Toast.makeText(this, "PDF 파일을 보기 위한 뷰어 앱이 없습니다.",
                        Toast.LENGTH_LONG).show();
            }
        } else {
            Toast.makeText(this, "PDF 파일이 없습니다.", Toast.LENGTH_LONG).show();
        }
    }
}
```

- **AndroidManifest.xml**

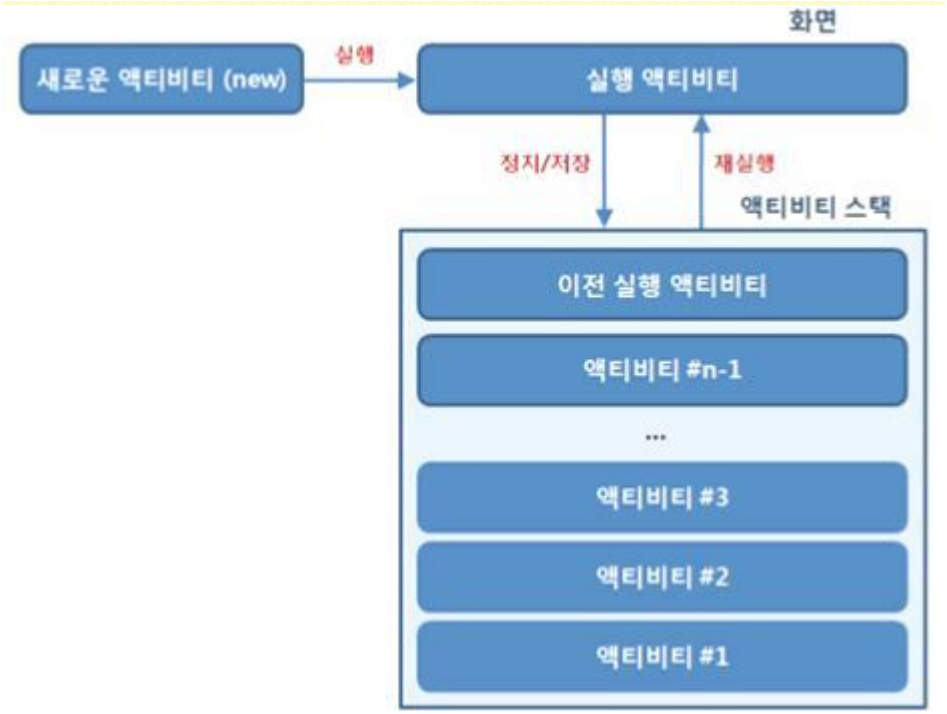
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.samplepdfview">

    <!-- SD 카드 접근 권한 추가 -->
    <!-- READ_EXTERNAL_STORAGE : SD 카드에서 파일을 읽어올 때 필요한 권한 -->
    <!-- WRITE_EXTERNAL_STORAGE : SD 카드에서 파일을 쓸 때 필요한 권한 -->
    <user-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <user-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

03-4. 액티비티를 위한 플래그와 부가 데이터

: 화면이 한 번 메모리에 만들어졌는데도 계속 액티비티 실행 메소드가 반복적으로 호출되면 동일한 액티비티가 여러 개 만들어집니다. 이때는 **플래그(Flag)**를 사용해 액티비티가 동작하는 방식을 조정할 수 있습니다.

플래그

- **액티비티 스택(Activity Stack)** : 액티비티를 차곡차곡 쌓아두었다가 가장 상위에 있던 액티비티가 없어지면 이전의 액티비티가 다시 화면에 보여진다.



- **대표적인 플래그들**
  - **FLAG_ACTIVITY_SINGLE_TOP** : 이미 생성된 액티비티가 있으면 그 액티비티를 그대로 사용하라는 플래그.
    - **getIntent() 메소드** : 부모 액티비티로부터 전달하는 인텐트는 새로 만들어진 인텐트의 onCreate() 메소드 안에서 getIntent() 메소드로 참조할 수 있다.
    - **onNewIntent() 메소드** : 액티비티가 새로 만들어지지 않았을 때 인텐트 객체만 전달 받을 수 있다.
  - **FLAG_ACTIVITY_NO_HISTORY** : 처음 이후에 실행된 액티비티는 액티비티 스택에 추가되지 않는다. 이 플래스는 알람 이벤트가 발생하여 사용자에게 한 번 알림 화면을 보여주고 싶을 때 유용하다.
  - **FLAG_ACTIVITY_CLEAR_TOP** : 액티비티 위에 있는 다른 액티비티를 모두 종료시키게 된다. 홈 화면과 같은 항상 우선하는 액티비티를 만들 때 유용하다.

부가 데이터

: 한 화면에서 다른 화면을 띄울 때 데이터를 전달해야만 하는 경우. 문자열이나 정수와 같은 부가 데이터를 넣을 때는 **키(Key)와 값(Value)**을 쌍으로 만들어 넣는다.

- **putExtra()** : 데이터를 넣는다.
- **get___Extra()** : 데이터를 가져온다.
  - **ex) putExtra(), getStringExtra()**
  - **참고**
    - Intent putExtra(String name, 자료형 value)
    - 자료형 get(자료형)Extra(자료형 name)
- 객체를 전달할 때는 **Parcelable 인터페이스**를 구현하여 전달한다.
  - **참고**
    - public abstract int describeContents()
    - public abstract void writeToParcel(Parcel dest, int flags)

      : 객체가 가지고 있는 데이터를 Parcle 객체로 만들어 주는 역할. 이 객체는 상수로 정의되므로 반드시 static final로 선언되어야 한다.

예제(Parcel 객체를 이용한 액티비티 간 데이터 전달)

- **SimpleData.java**

```
package com.example.sampleparcelable;

import android.os.Parcel;
import android.os.Parcelable;

public class SimpleData implements Parcelable {
    int number;
    String message;

    public SimpleData(int num, String msg) {
        number = num;
        message = msg;
    }

    // Parcel 객체에서 읽기
    public SimpleData(Parcel src) {
        number = src.readInt();
        message = src.readString();
    }

    // CREATOR 상수 정의
    public static final Parcelable.Creator CREATOR = new Parcelable.Creator() {
        public SimpleData createFromParcel(Parcel in) {
            // SimpleData 생성자를 호출해 Parcel 객체에서 읽기
            return new SimpleData(in);
        }
```

```
        public SimpleData[] newArray(int size) {
            return new SimpleData[size];
        }
    };


    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    // Parcel 객체로 쓰기
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeInt(number);
        dest.writeString(message);
    }

    public int getNumber() {
        return number;
    }

    public String getMessage() {
        return message;
    }
}
```

- **activity_main.xml**

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

    <Button
        android:id="@+id/button"
        android:onClick="onButton1Clicked"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="메뉴화면 띄우기"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

- **MainActivity.java**

```
package com.example.sampleparcelable;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    public static final int REQUEST_CODE_MENU = 101;
    public static final String KEY_SIMPLE_DATA = "data";

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

    public void onButton1Clicked(View v) {
        // 인텐트 객체 생성
        Intent intent = new Intent(getApplicationContext(), MenuActivity.class);

        // SimpleData 객체 생성
        SimpleData data = new SimpleData(100, "Hello Android!");

        // 인텐트에 부가 데이터로 넣기
        intent.putExtra(KEY_SIMPLE_DATA, data);

        startActivityForResult(intent, REQUEST_CODE_MENU);
    }
}
```

- **activity_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MenuActivity">

    <TextView
```

```
        android:id="@+id/textView"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:text="전달받은 데이터"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/button2"
        android:text="돌아가기"
        android:onClick="onButton2Clicked"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</android.support.constraint.ConstraintLayout>
```

- **MenuActivity.java**

```
package com.example.sampleparcelable;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MenuActivity extends AppCompatActivity {
    TextView textView;

    public static final String KEY_SIMPLE_DATA = "data";

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

        textView = (TextView)findViewById(R.id.textView);
        Button button = (Button)findViewById(R.id.button2);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // 인텐트 객체를 만든다.
                Intent intent = new Intent();
                intent.putExtra("name", "mike");

                // 응답을 전달하고 이 액티비티를 종료한다.
```

```
                setResult(RESULT_OK, intent);
                finish();
            }
        });

        // 메인 액티비티로부터 전달 받은 인텐트를 확인한다.
        Intent intent = getIntent();
        processIntent(intent);
    }

    private void processIntent(Intent intent) {
        if (intent != null) {
            // 인텐트 안의 번들 객체를 참조한다.
            Bundle bundle = intent.getExtras();

            // 번들 객체 안의 SimpleData 객체를 참조한다.
            SimpleData data = (SimpleData) bundle.getParcelable(KEY_SIMPLE_DATA);

            // 텍스트뷰에 값을 보여준다.
            textView.setText("전달 받은 데이터\nNumber : " + data.getNumber() +
"\nMessage : " + data.getMessage());
        }
    }
}
```

- **실행 결과**

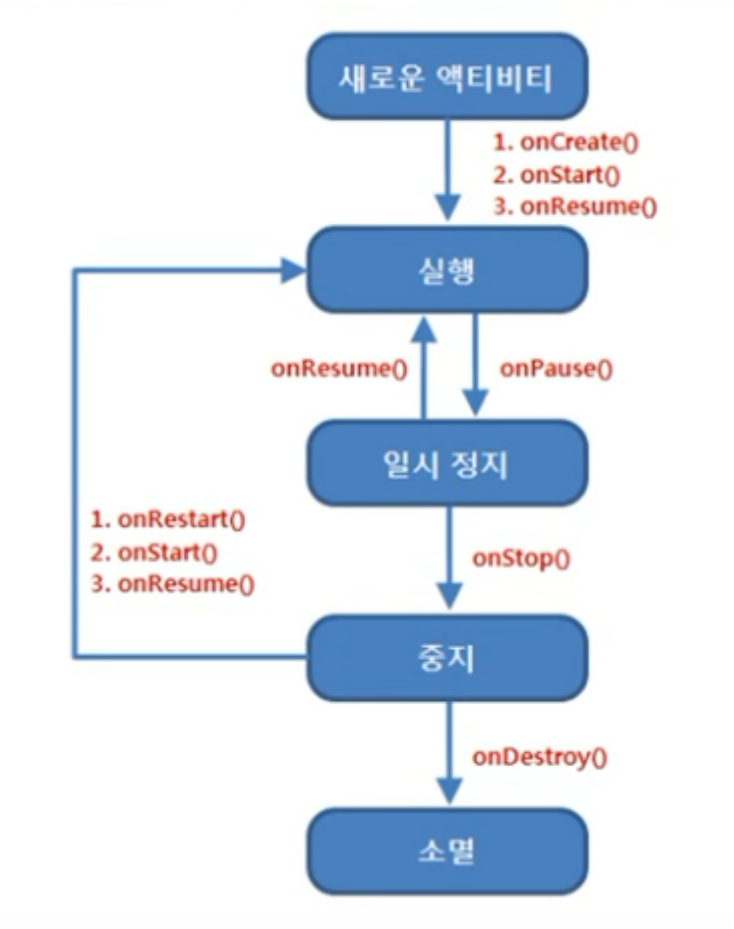
메뉴화면 띄우기 클릭시

03-5. 액티비티의 수명주기

: 액티비티는 처음 실행될 때 메모리에 만들어지는 과정부터 시작해 실행과 중지 그리고 메모리에서 해체되는 여러 과정을 상태 정보로 가지고 있게 되고 이러한 상태 정보는 시스템에서 관리하면서 각각의 상태에 해당하는 메소드를 자동으로 호출하게 된다.

- **대표적인 상태 정보(수명주기)**

상 태	설 명
실행(Running)	화면 상에 액티비티가 보이면서 실행되어 있는 상태
일시 중지(Paused)	다른 액티비티가 위에 있어 포커스를 받지 못하는 상태
중지(Stopped)	다른 액티비티에 의해 완전히 가려져 보이지 않는 상태



1. **새로운 액티비티**

   : onCreate(), onStart(), onResume() 메소드가 차례대로 호출

2. **실행**

   : 다른 액티비티가 상위에 오게 되면 onPause() 메소드 호출(일시정지).

3. **일시정지**

   : 중지 상태로 변경될 때 자동으로 onStop() 메소드 호출(중지). 액티비티가 다시 실행될 때는 onResume() 메소드 호출(실행)

4. **중지**

   : 액티비티가 다시 실행될 때는 onResume() 메소드 호출(실행). 만약 액티비티가 메모리 상에서 없어질 경우에는 onDestroy() 메소드 호출(소멸).

- **onSaveInstanceState() 메소드** : 데이터를 임시로 저장하는 메소드

- **onRestoreInstanceState() 메소드** : 저장했던 데이터 전달 메소드

예제(토스트 메시지를 넣어 수명주기 확인하기)

- **activity_main.xml**

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


    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="onButton1Clicked"
        android:text="메뉴화면 띄우기"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/nameInput"
        android:layout_marginTop="200dp"
        app:layout_constraintBottom_toTopOf="@id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:hint="이름을 입력하세요."
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</android.support.constraint.ConstraintLayout>
```

- **activity_menu.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MenuActivity">

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="onButton2Clicked"
        android:text="돌아가기"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

- **MainActivity.java**

```
package com.example.samplelifecycle;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    EditText nameInput;

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

        nameInput = (EditText)findViewById(R.id.nameInput);
        Toast.makeText(this, "onCreate 호출됨", Toast.LENGTH_LONG).show();
    }
```

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

        Toast.makeText(this, "onStart 호출됨", Toast.LENGTH_LONG).show();
    }

    @Override
    protected void onStop() {
        super.onStop();

        Toast.makeText(this, "onStop 호출됨", Toast.LENGTH_LONG).show();
    }

    // 앱이 갑자기 중지되거나 또는 종료되면 데이터가 사라지기 때문에
    // onPause 메소드 안에서 데이터를 저장하고
    // onResume 메소드 안에서 복원해야 한다.
    // 앱 안에서 간단한 데이터를 저장하거나 복원할 때는 SharedPreferences를 사용할 수 있다.
    // 이것은 앱 내부에 파일을 하나 만들고 이 파일 안에서 데이터를 저장하거나 읽어올 수 있도록 한
다.

    @Override
    protected void onResume() {
        super.onResume();

        Toast.makeText(this, "onResume 호출됨", Toast.LENGTH_LONG).show();

        // 설정 정보에 저장된 데이터를 복원
        restoreState();
    }

    @Override
    protected void onPause() {
        super.onPause();

        Toast.makeText(this, "onPause 호출됨", Toast.LENGTH_LONG).show();

        // 현재 입력상자에 입력된 데이터 저장
        saveState();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        Toast.makeText(this, "onDestroy 호출됨", Toast.LENGTH_LONG).show();
        clearMyPrefs();
    }

    public void onButton1Clicked(View v) {
        Intent intent = new Intent(getApplicationContext(), MenuActivity.class);
        startActivity(intent);
```

```
    }

    protected void restoreState() {
        SharedPreferences pref = getSharedPreferences("pref",
Activity.MODE_PRIVATE);

        if ((pref != null) && (pref.contains("name")) ) {
            String name = pref.getString("name", "");
            nameInput.setText(name);
        }
    }
    protected void saveState() {
        SharedPreferences pref = getSharedPreferences("pref",
Activity.MODE_PRIVATE);
        SharedPreferences.Editor editor = pref.edit();
        editor.putString("name", nameInput.getText().toString());
        editor.commit();
    }

    protected void clearMyPrefs() {
        SharedPreferences pref = getSharedPreferences("pref",
Activity.MODE_PRIVATE);
        SharedPreferences.Editor editor = pref.edit();
        editor.clear();
        editor.commit();
    }
}
```

- **MenuActivity.java**

```
package com.example.samplelifecycle;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MenuActivity extends AppCompatActivity {

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

    public void onButton2Clicked(View v) {
        finish();
    }
}
```

03-6. 서비스

: 백그라운드에서 실행되는 프로세스를 의미한다. 새로 만든 서비스는 항상 매니페스트 파일에 등록해야 한다. 그리고 서비스를 실행시키고 싶다면 메인 액티비티에서 startService() 메소드를 호출하면 된다.

- 서비스는 화면이 없는 상태에서 백그라운드로 실행됨
- 서비스는 프로세스가 종료되어도 시스템에서 자동으로 재시작함

- **startService()** : 인텐트를 전달하는 목적으로 많이 쓰인다
- **onStartCommand()** : 전달 받은 인텐트를 처리하게 된다.

예제(서비스로부터 전달 받은 인탠트를 Log 출력 및 액티비티에서의 처리)

- **activity_main.xml**

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

    <EditText
        android:id="@+id/editText"
        android:layout_width="300dp"
```

```
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginBottom="8dp"
            android:text="김진수"
            app:layout_constraintBottom_toTopOf="@+id/button"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginLeft="8dp"
            android:layout_marginTop="8dp"
            android:layout_marginEnd="8dp"
            android:layout_marginRight="8dp"
            android:layout_marginBottom="8dp"
            android:onClick="onButton1Clicked"
            android:text="서비스로 보내기"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />


    </android.support.constraint.ConstraintLayout>
```

- **MainActivity.java**

```
    package com.example.sampleservice;

    import android.content.Intent;
    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.EditText;
    import android.widget.Toast;

    public class MainActivity extends AppCompatActivity {
        EditText editText;

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

            editText = (EditText)findViewById(R.id.editText);   // 입력 상자 참조
```

```
        // getIntent() 메소드를 호출하여 인텐트 객체를 참조한다.
        Intent passedIntent = getIntent();
        processIntent(passedIntent);
    }

    // 만약 MainActivity 가 메모리에 만들어져 있다면
    // onNewIntent() 메소드로 전달된다.
    protected void onNewIntent(Intent intent) {
        // processIntent() 메소드를 만들고 그 안에서 객체 처리
        processIntent(intent);

        super.onNewIntent(intent);
    }

    private void processIntent(Intent intent) {
        if (intent != null) {
            String command = intent.getStringExtra("command");
            String name = intent.getStringExtra("name");

            // 인텐트로 전달 받은 데이터를 토스트 메시지로 출력
            Toast.makeText(this, "command : " + command + " , name : " + name,
                    Toast.LENGTH_LONG).show();
        }
    }

    public void onButton1Clicked(View v) {
        String name = editText.getText().toString();        // 입력 상자 내용 가져옴

        // 인텐트 객체 생성
        Intent intent = new Intent(this, MyService.class);

        // 두 개의 부가 데이터
        // 서비스 쪽으로 전달한 인텐트 객체의 데이터가 어떤 목적으로 사용되는지 구별하기 위함
        intent.putExtra("command", "show");

        // 입력상자에서 가져온 문자열을 전달하기 위함
        intent.putExtra("name", name);
        startService(intent);
    }
}
```

- **MyService.java**

```
package com.example.sampleservice;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class MyService extends Service {
    public static final String TAG = "MyService";
```

```
public MyService() {
}

@Override
public IBinder onBind(Intent intent) {
    // TODO: Return the communication channel to the service.
    throw new UnsupportedOperationException("Not yet implemented");
}

// Log.d() 메소드를 통해서 로그를 출력시킨다.
// Log의 첫 번째 파라미터로 태그 문자열(로그를 구별하는 역할)을 전달해야 한다.
@Override
public void onCreate() {
    super.onCreate();

    Log.d(TAG, "onCreate() 호출됨.");
}

@Override
// 인텐트 객체를 전달 받는 중요한 메소드, 시스템에 의해 자동으로 다시 시작된다.
public int onStartCommand(Intent intent, int flags, int startId) {
    Log.d(TAG, "onStartCommand() 호출됨");

    // 인텐트 객체가 null 인지 먼저 체크
    if (intent == null) {
        // 이 값을 반환하면 서비스가 비정상 종료되었을 때 시스템이 자동으로 재시작한다.
        return Service.START_STICKY;
    } else {
        // 코드를 너무 많이 넣으면 복잡하므로
        // 메소드를 새롭게 정의한 후 호출한다.
        processCommand(intent);
    }

    return super.onStartCommand(intent, flags, startId);
}

@Override
public void onDestroy() {
    super.onDestroy();
}

private void processCommand(Intent intent) {
    String command = intent.getStringExtra("command");
    String name = intent.getStringExtra("name");

    Log.d(TAG, "command : " + command + ", name : " + name );

    // 5초 동안 1초에 한 번씩 로그를 출력시킨다.
    for (int i = 0; i < 5; i++) {
        try {
            Thread.sleep(1000);
        } catch (Exception e) { };
```

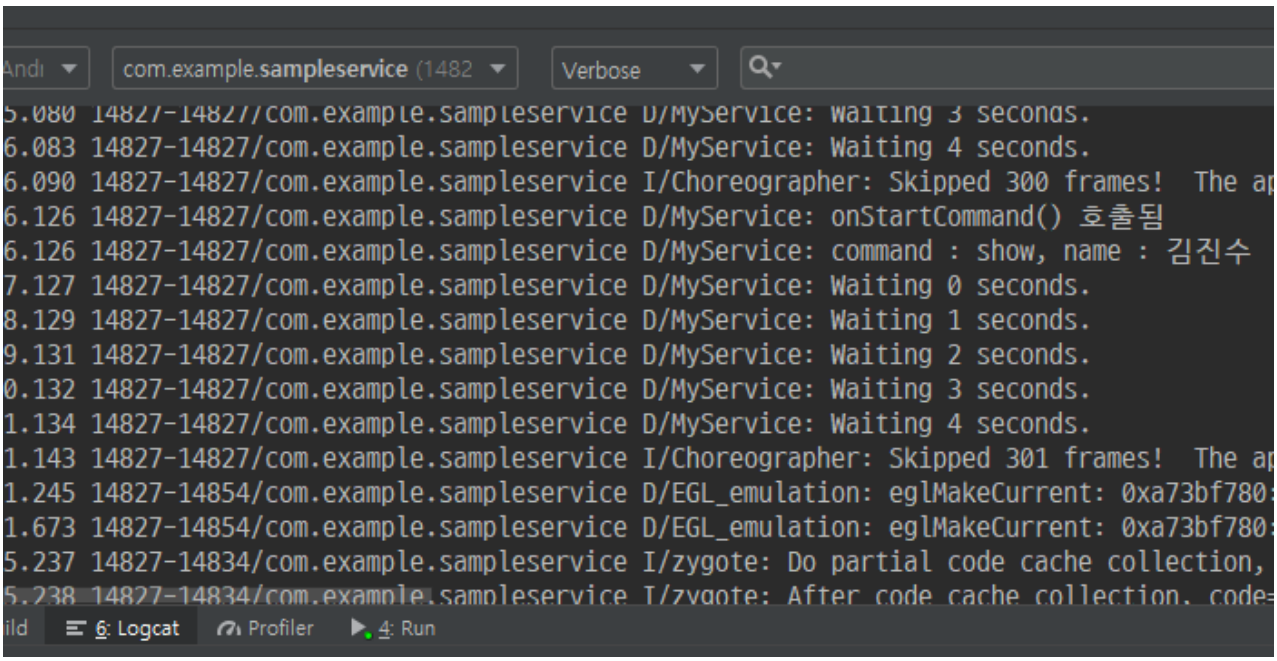
```
            Log.d(TAG, "Waiting " + i + " seconds.");
        }

        // 인텐트 객체를 new 연산자로 생성
        // 첫 번째 파라미터로 getApplicationContext() 메소드 호출하여 Context 객체 전달
        // 두 번째 파라미터로 MainActivity.class 객체 전달
        Intent showIntent = new Intent(getApplicationContext(), MainActivity.class);

        // 인텐트 객체에 Flags 추가
        // 새로운 태스크를 생성하도록 FLAG_ACTIVITY_NEW_TASK 추가
        // 객체가 이미 만들어져 있을 때 재사용하도록 FLAG_ACTIVITY_SINGLE_TOP 과
        // FLAG_ACTIVITY_CLEAR_TOP 플래그 추가
        showIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
                Intent.FLAG_ACTIVITY_SINGLE_TOP |
                intent.FLAG_ACTIVITY_CLEAR_TOP);

        // 인텐트 객체에 부가 데이트 추가
        showIntent.putExtra("command", "show");
        showIntent.putExtra("name", name + " from service.");
        startActivity(showIntent);
    }
}
```

- **실행 결과**

버튼 클릭시


3:34
SampleService
김진수
서비스로 보내기
command : show , name : 김진수 from service.

Logcat 화면



03-7. 브로드캐스트 수신자

- **브로드캐스팅(Broadcating)** : 메시지를 여러 객체에게 전달하는 것. 안드로이드에서는 여러 애플리케이션 구성 요소에게 메시지를 전달하고 싶을 경우 브로드캐스팅을 사용한다(글로벌 이벤트). 브로드캐스팅 메시지를 받고 싶다면 **브로드캐스트 수신자(Broadcast Receiver)**를 만들어 등록하면 된다. 브로드캐스트 수신자를 만들게 되면 매니페스트 파일에 등록해야 한다.

예제(브로드캐스트 수신자로 SMS 문자 가져오기)

- **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.samplereceiver">
    <!-- SMS 수신할 때 필요한 권한 추가 -->
    <uses-permission android:name="android.permission.RECEIVE_SMS" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".SmsActivity"></activity>

        <!-- 브로드캐스트 리시버 추가 -->
```

```
        <receiver
            android:name=".SmsReceiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

- **SmsReceiver.java**

```
package com.example.samplereceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;

import java.text.SimpleDateFormat;
import java.util.Date;

public class SmsReceiver extends BroadcastReceiver {
    public static final String TAG = "SmsReceiver";
    public SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

    @Override
    // SMS를 받으면 onReceive() 메소드가 자동으로 호출
    // 파라미터로 전달되는 Intent 객체 안에 SMS 데이터가 들어 있다.
    public void onReceive(Context context, Intent intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        Log.i(TAG, "onReceive() 메소드 호출됨.");

        // 인텐트 안에 들어 있는 SMS 메시지를 파싱한다.
        // 인텐트 객체 안에 있는 Bundle 객체를 getExtras() 메소드로 참조한다.
        Bundle bundle = intent.getExtras();

        // parseSmsMessage() 메소드로 SMS 메시지 객체를 만든다.
```

```
        SmsMessage[] messages = parseSmsMessage(bundle);

        if (messages != null && messages.length > 0) {
            // SMS 발신 번호 확인
            // 발신자 번호를 확인하기 위해 getOriginaingAddress() 메소드 호출
            String sender = messages[0].getOriginatingAddress();
            Log.i(TAG, "SMS contents : " + sender);

            // SMS 메시지 확인
            // 문자 내용을 확인하기 위해 getMessageBody() 메소드 호출
            String contents = messages[0].getMessageBody();

            Log.i(TAG, "SMS contents : " + contents);

            // SMS 수신 기간 확인
            // 문자를 받은 시각을 확인하기 위해 getTimestampMillis() 메소드 호출
            Date receivedDate = new Date(messages[0].getTimestampMillis());

            Log.i(TAG, "SMS received date : " + receivedDate.toString());

            sendToActivity(context, sender, contents, receivedDate);
        }
    }

    // SmsActivity 로 인텐트를 보내기 위해 만든 메소드
    private void sendToActivity(Context context, String sender, String contents,
                                Date receivedDate) {

        // 메시지를 보여줄 액티비티를 띄운다.
        // Intent 객체를 만들 때 두 번째 파라미터로 SmsActivity.class 객체를
        // 전달했으므로 startActivity() 메소드를 사용해 이 인텐트를
        // 시스템으로 전달하면 시스템이 그 인텐트를 SmsActivity  쪽으로 전달한다.
        Intent myIntent = new Intent(context, SmsActivity.class);

        // 플래그를 이용한다.
        // 브로드캐스트 수신자는 화면이 없으므로 인텐트의 플래그로
        // FLAG_ACTIVITY_NEW_TASK 를 추가해야 한다는 점을 잊지 말자!!
        // 그리고 이미 메모리에 만든 SmsActivity 가 있을 때 추가로 만들지 않도록
        // FLAG_ACTIVITY_SINGLE_TOP 플래그도 추가한다.
        myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
                Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TOP);

        myIntent.putExtra("sender", sender);
        myIntent.putExtra("contents", contents);
        myIntent.putExtra("receivedDate", format.format(receivedDate));

        context.startActivity(myIntent);
    }

    // SMS 데이터를 확인할 수 있도록 만드는 안드로이드 API에 정해둔 코드
    private SmsMessage[] parseSmsMessage(Bundle bundle) {
        Object[] objs = (Object[]) bundle.get("pdus");
        SmsMessage[] messages = new SmsMessage[objs.length];
```

```
        int smsCount = objs.length;
        for (int i = 0; i < smsCount; i++) {
            // PDU 포맷으로 되어 있는 메시지를 복원합니다.
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                String format = bundle.getString("format");

                // SMS 데이터를 확인하기 위해 SmsMessage 클래스의 createFromPdu() 메소드를
사용해
                // SmsMessage 객체로 변환 후 SMS 데이터를 확인할 수 있다.
                messages[i] = SmsMessage.createFromPdu((byte[]) objs[i], format);
            } else {
                messages[i] = SmsMessage.createFromPdu((byte[]) objs[i]);
            }
        }

        return messages;
    }
}
```

- **activity_sms.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SmsActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <EditText
            android:id="@+id/editText"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:hint="발신번호" />
```

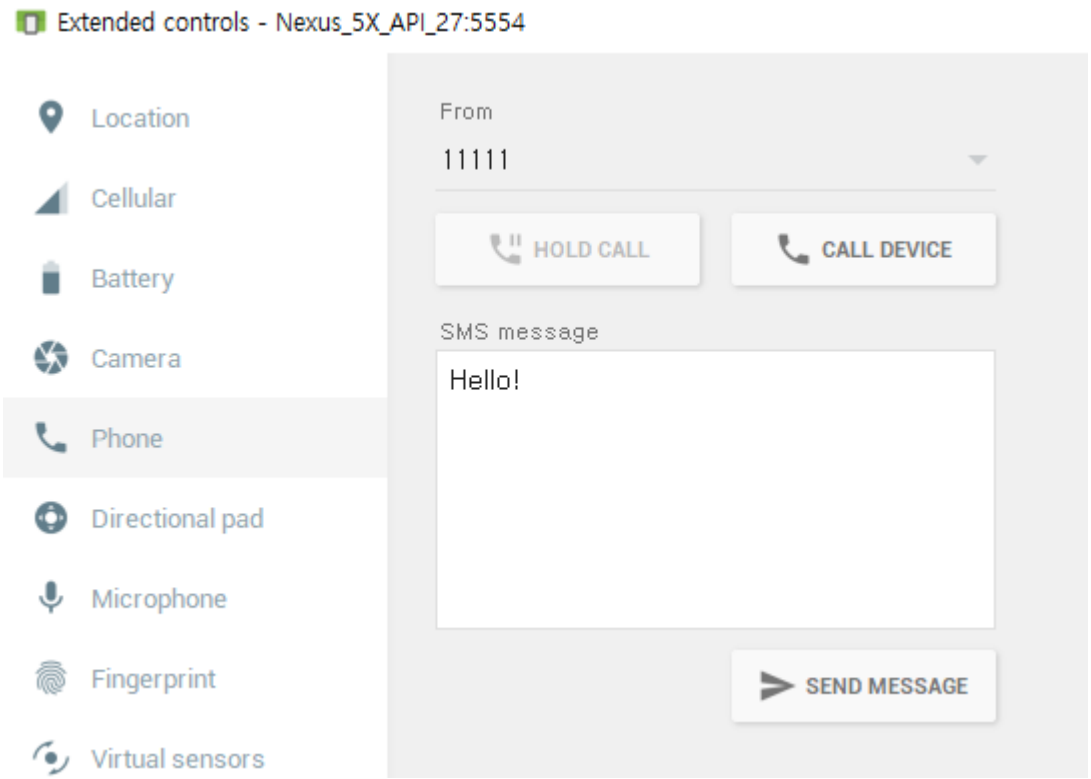
```
        <EditText
            android:id="@+id/editText2"
            android:layout_width="348dp"
            android:layout_height="389dp"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="10dp"
            android:hint="내용" />

        <EditText
            android:id="@+id/editText3"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:layout_marginTop="10dp"
            android:hint="수신시각" />

        <Button
            android:id="@+id/button"
            android:onClick="onButton1Clicked"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:text="확인" />

    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

- **SmsActivity.java**

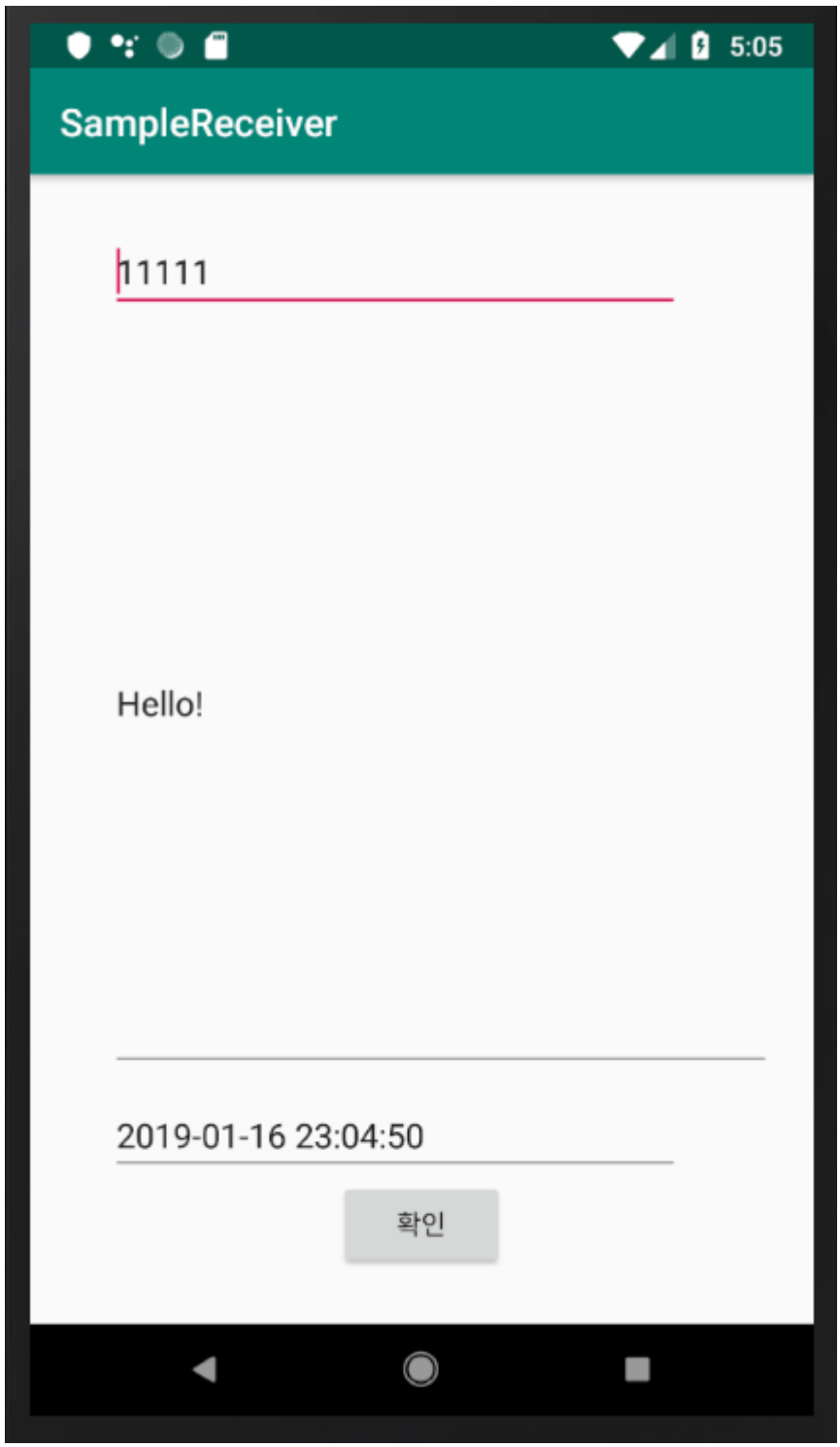
```
package com.example.samplereceiver;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class SmsActivity extends AppCompatActivity {
    EditText editText;
    EditText editText2;
    EditText editText3;

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

        editText = (EditText)findViewById(R.id.editText);
        editText2 = (EditText)findViewById(R.id.editText2);
        editText3 = (EditText)findViewById(R.id.editText3);
        Button button = (Button)findViewById(R.id.button);
```

```
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });

        Intent passedIntent = getIntent();
        processIntent(passedIntent);
    }

    @Override
    // 이 액티비티가 이미 만들어져 있는 상태에서 전달 받은 인탠트 처리하기 위한 메소드
    protected void onNewIntent(Intent intent) {
        processIntent(intent);

        super.onNewIntent(intent);
    }

    private void processIntent(Intent intent) {
        if (intent != null) {
            String sender = intent.getStringExtra("sender");
            String contents = intent.getStringExtra("contents");
            String receivedDate = intent.getStringExtra("receivedDate");

            editText.setText(sender);
            editText2.setText(contents);
            editText3.setText(receivedDate);
        }
    }
}
```

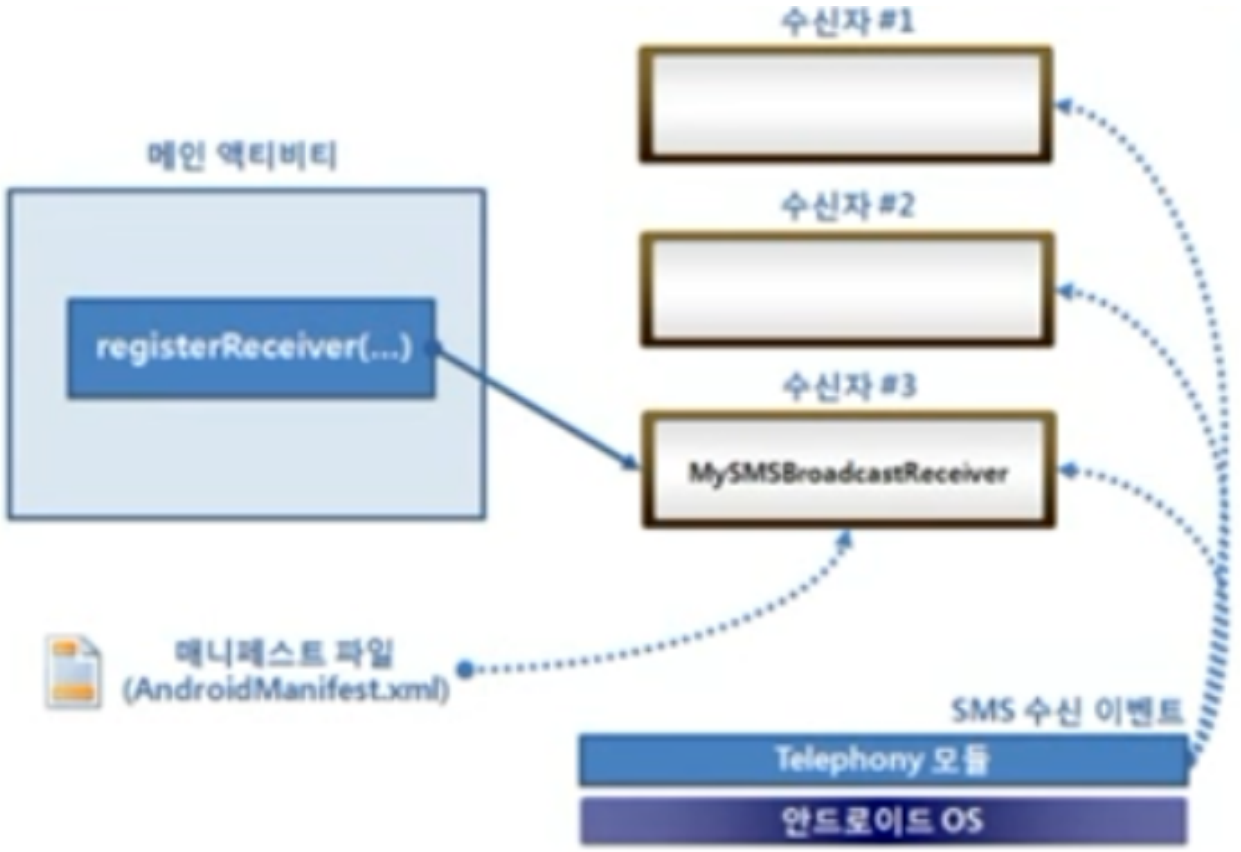
- **실행 결과**


Extended controls - Nexus_5X_API_27:5554
Location
Cellular
Battery
Camera
Phone
Directional pad
Microphone
Fingerprint
Virtual sensors
From
11111
HOLD CALL
CALL DEVICE
SMS message
Hello!
SEND MESSAGE

가상으로 문자를 보낸다.



> 브로드 캐스트 수신자는 매니페스트 파일안에 <receiver> 태그로 추가되어 있지만 매니페스트에 등록하지 않고 자바 소스 파일에서 registerReceiver() 메소드를 사용해 등록할 수도 있다. 만약 다른 브로드캐스트 수신자에게 메시지를 보내고 싶은 경우에는 sendBroadcast() 메소드를 사용할 수 있다.



03-8. 위험 권환 부여하기

: 일반 권한과 위험 권한으로 나뉜다. 앱을 실행할 때 사용자로부터 권한을 부여받도록 변경 되었다.

- **위험 권한의 세부 정보**

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

예제(SMS 위험권한 확인하기 및 권한 권한 부여 요청)

- **MainActivity.java**

```
package com.example.samplereceiver;

import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {

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

        // 권한이 이미 부여되어 있을 수도 있으므로 권한 확인 메소드 호출
        int permissionCheck = ContextCompat.checkSelfPermission(this,
Manifest.permission.RECEIVE_SMS);

        // 권한이 부여 되어 있는지 확인함
        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 {
                // 사용자가 볼 수 있도록 새로운 권한 부여
                // 요청 대화 상자를 띄움
                ActivityCompat.requestPermissions(this,
                        new String[] { Manifest.permission.RECEIVE_SMS}, 1);
            }
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        switch (requestCode) {
            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;
            }
        }
    }
}
```