안드로이드 앱 프로그래밍

Chapter 05

여러 화면 간 전환하기

이번 장에서는 무엇을 다룰까요?

여러 화면들이 있는 제대로 된 앱을 만들고 싶어요.

- XML로 만든 레이아웃이 어떻게 화면에 보여지는 것일까요?
- 여러 화면들을 만들고 나면 어떻게 화면을 전환시킬까요?
- 화면이 바뀔 때 데이터는 어떻게 전달할까요?

이번 장에서는 무엇을 다룰까요?

XML로 만든 레이아웃은 어떻게 화면에 보여지는 것일까요?	• 레이아웃 인플레이션 이해하기
화면을 더 추가하고 싶어요	• 화면 구성과 화면 간 전환 • 인텐트 살펴보기
다른 화면으로 데이터를 전달할 수 있나요? 태스크라는 것은 뭔가요?	• 액티비티를 위한 플래그와 부가 데이터 • 태스크 관리 이해하기
수명주기에 대해 알고 싶어요	• 액티비티의 수명주기
간단한 데이터를 저장했다가 가져오려면 어떻게 하나요?	• SharedPrefernaces 사용하기

강의 주제 및 목차

강의 주제

화면을 여러 개 만들고 화면 간에 전환하는 방법에 대한 이해

1. 레이아웃 인플레이션 이해하기
2. 여러 화면 만들고 화면 간 전환하기
3. 인텐트 살펴보기
4. 플래그와 부가 데이터 사용하기
5. 태스크 관리 이해하기
6. 액티비티의 수명주기와 SharedPreferences 이해하기

1. 레이아웃 인플레이션 이해하기

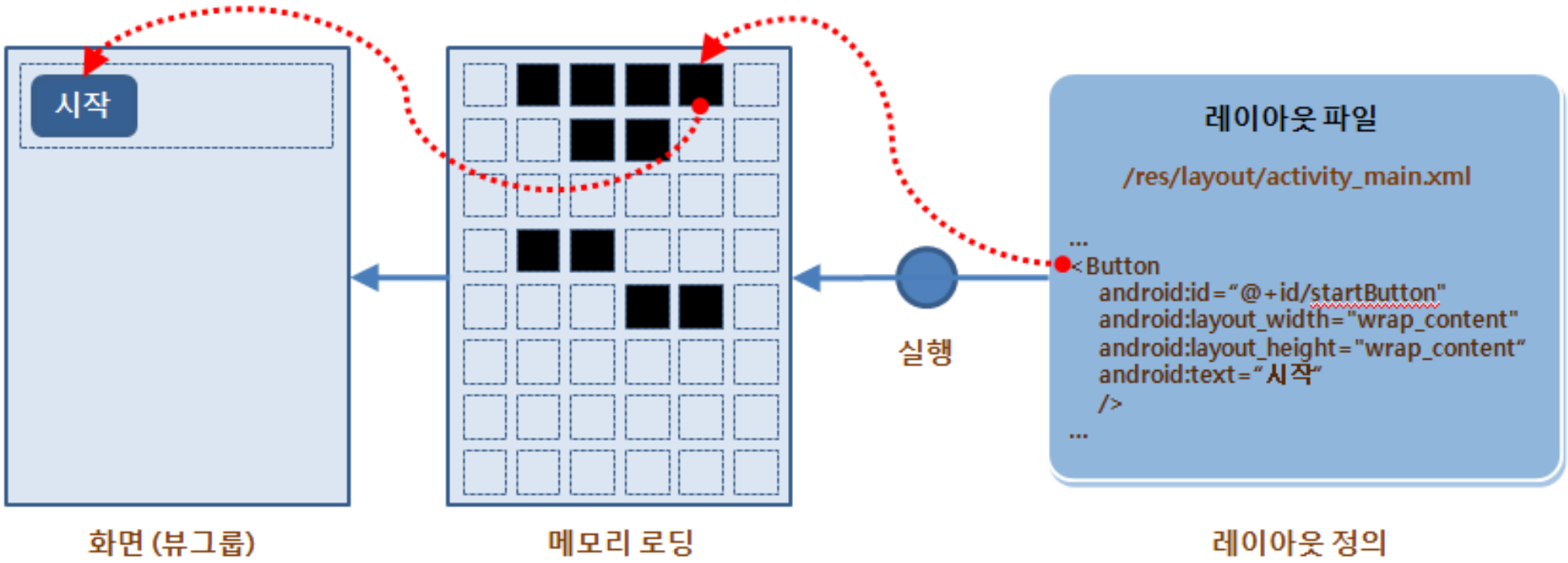
XML 레이아웃 파일과 자바 소스 파일의 매칭

- setContentView 메소드에서 XML 레이아웃 파일 매칭

R.layout.레이아웃 파일 이름

인플레이션이란?

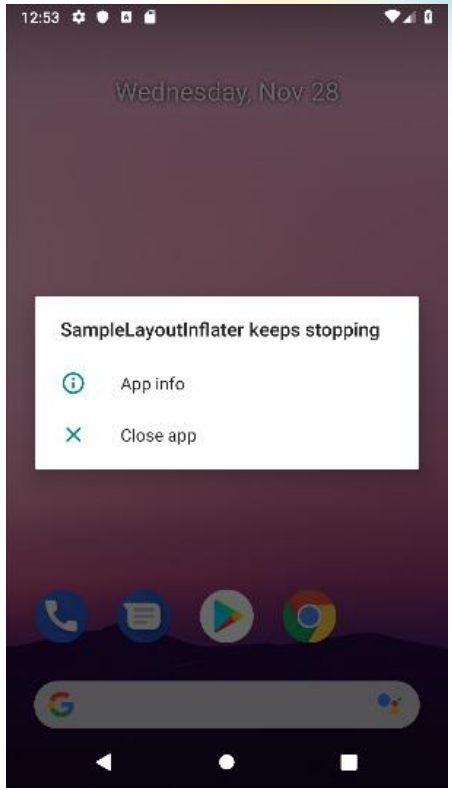
- 인플레이션 : XML 레이아웃에 정의된 내용이 메모리에 객체화되는 과정



["시작" 버튼의 레이아웃 인플레이션 과정]

레이아웃 인플레이션의 이해 – 호출 순서

```
중략…

public class MainActivity extends AppCompatActivity {

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

        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();
            }
        });

        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])

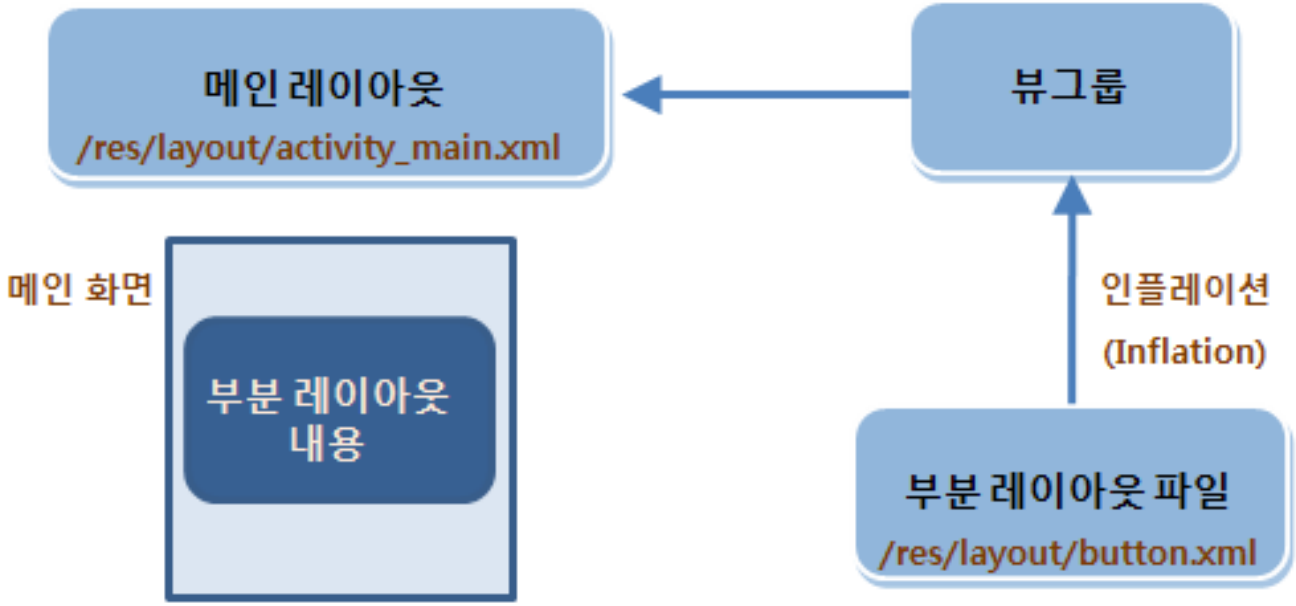
- **setContentView() 메소드의 역할**
  - 화면에 나타낼 뷰를 지정하는 역할
  - XML 레이아웃의 내용을 메모리 상에 객체화하는 역할

[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

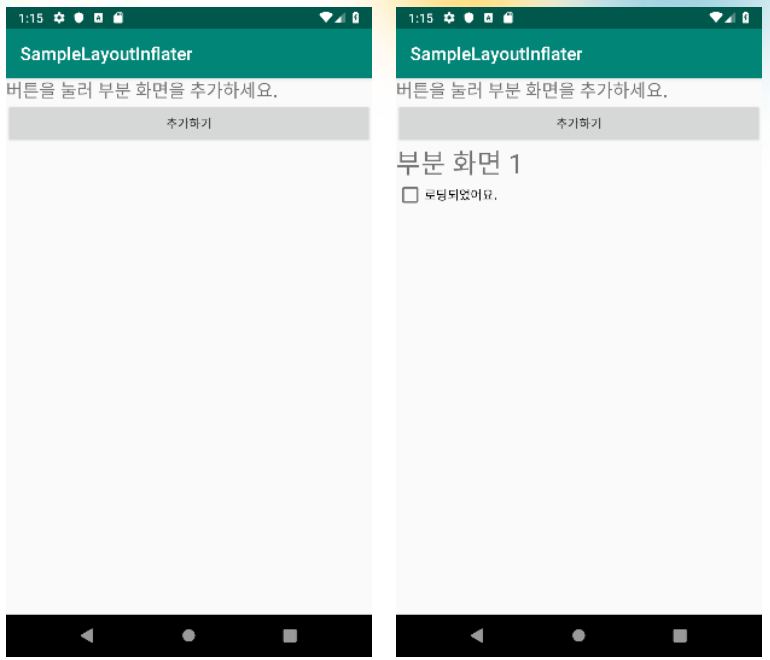
- **부분 화면 → 수동으로 인플레이션**
  - 부분화면을 위한 자바 소스 1개 또는 뷰 (뷰가 1개의 소스 파일로 분리될 수 있음)
  - 부분화면을 위한 XML 레이아웃 1개 : singer.xml

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

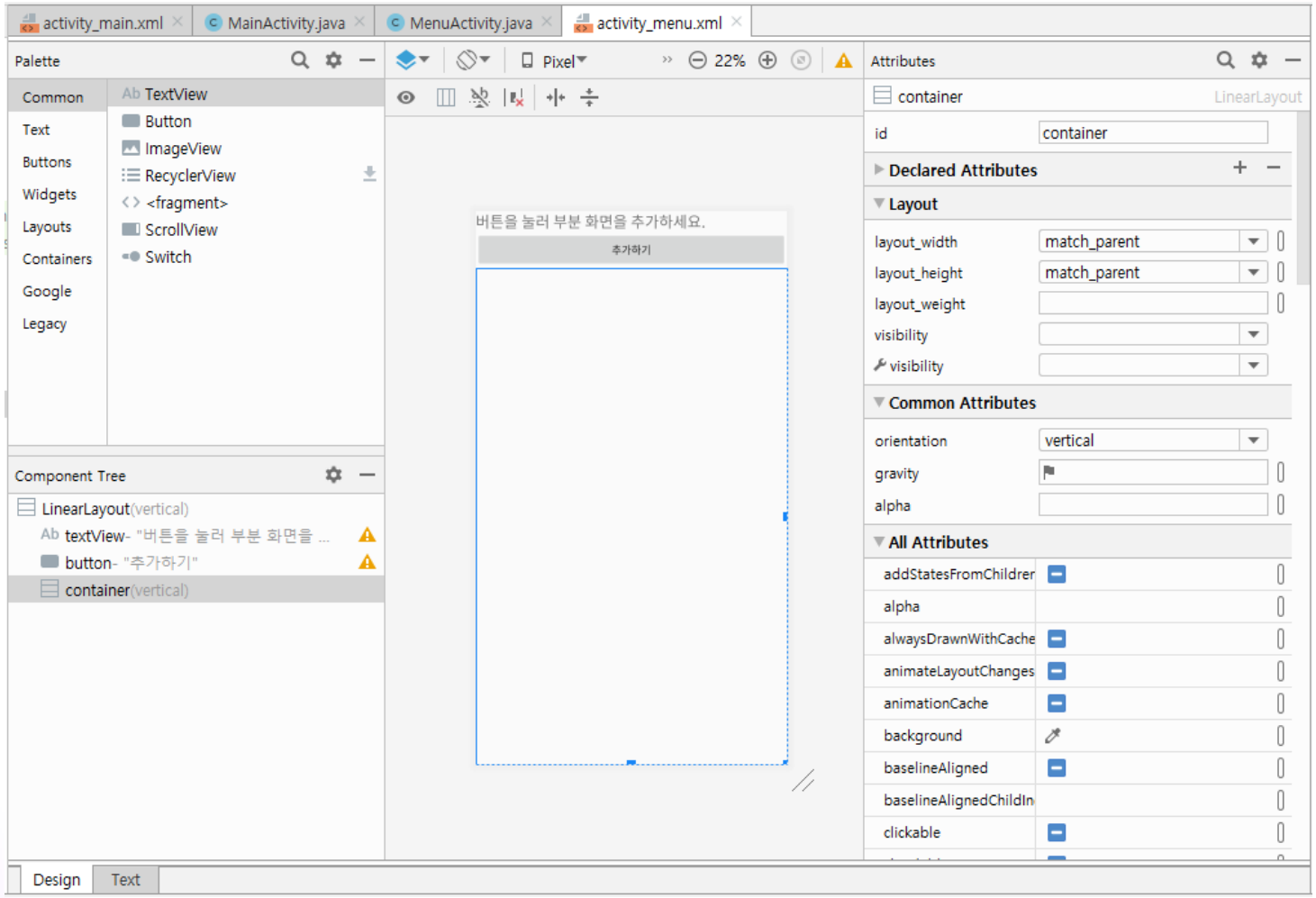
-레이아웃 코드 작성

메인 액티비티 코드

-메인 액티비티 코드 작성

메인 액티비티의 레이아웃

- 위쪽에 버튼을 배치하고 아래쪽에 다른 레이아웃이 들어갈 공간을 확보

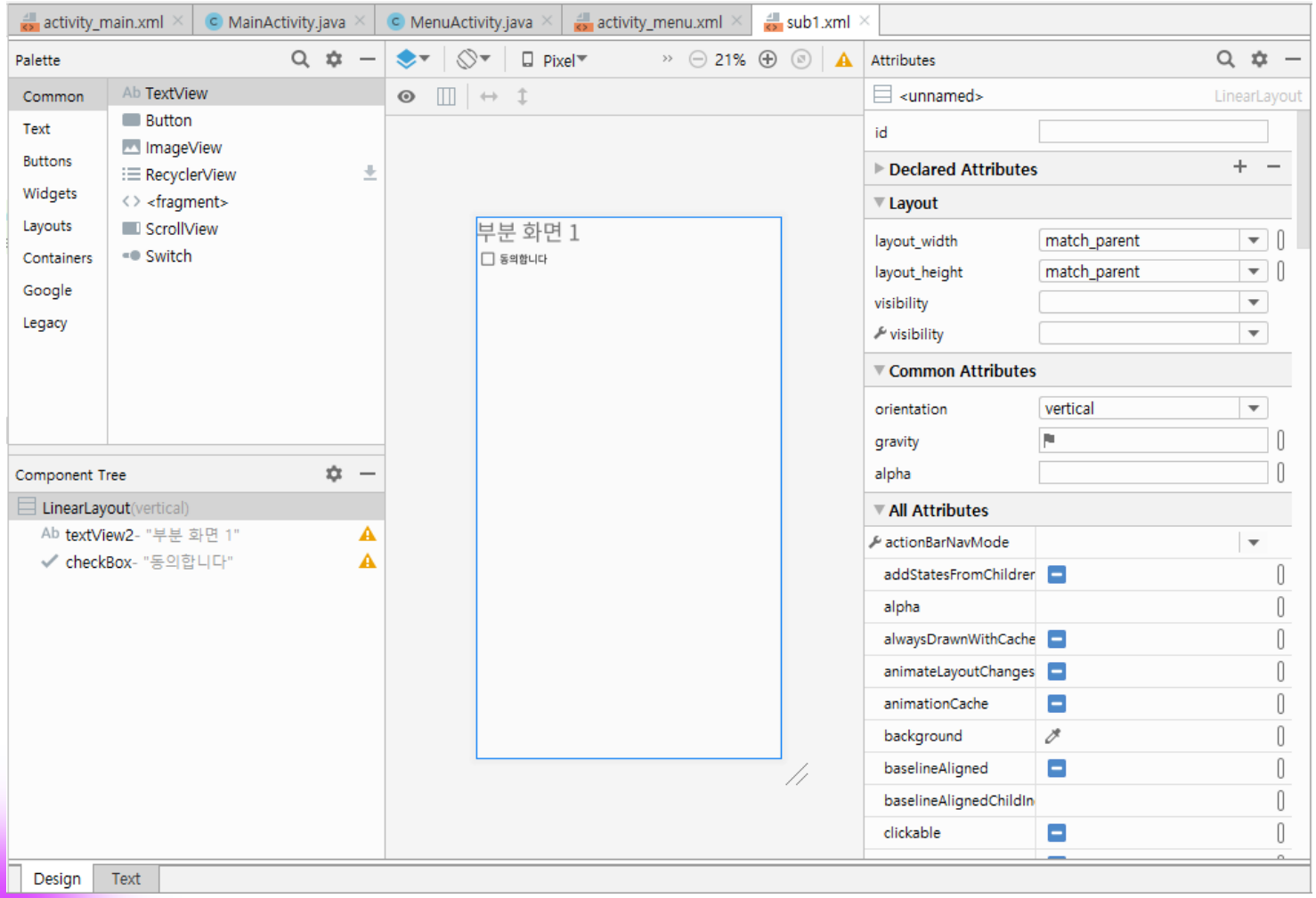


메인 액티비티의 레이아웃

```
<?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>
```

부분 화면을 위한 레이아웃

- sub1.xml 파일로 만들고 텍스트뷰와 체크박스 위젯 추가



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.");
    }
  });
}
```

레이아웃 인플레이션 메소드

[Reference]

View inflate (int resource, ViewGroup root)

[Reference]

static LayoutInflater LayoutInflater.from (Context context)

[Reference]

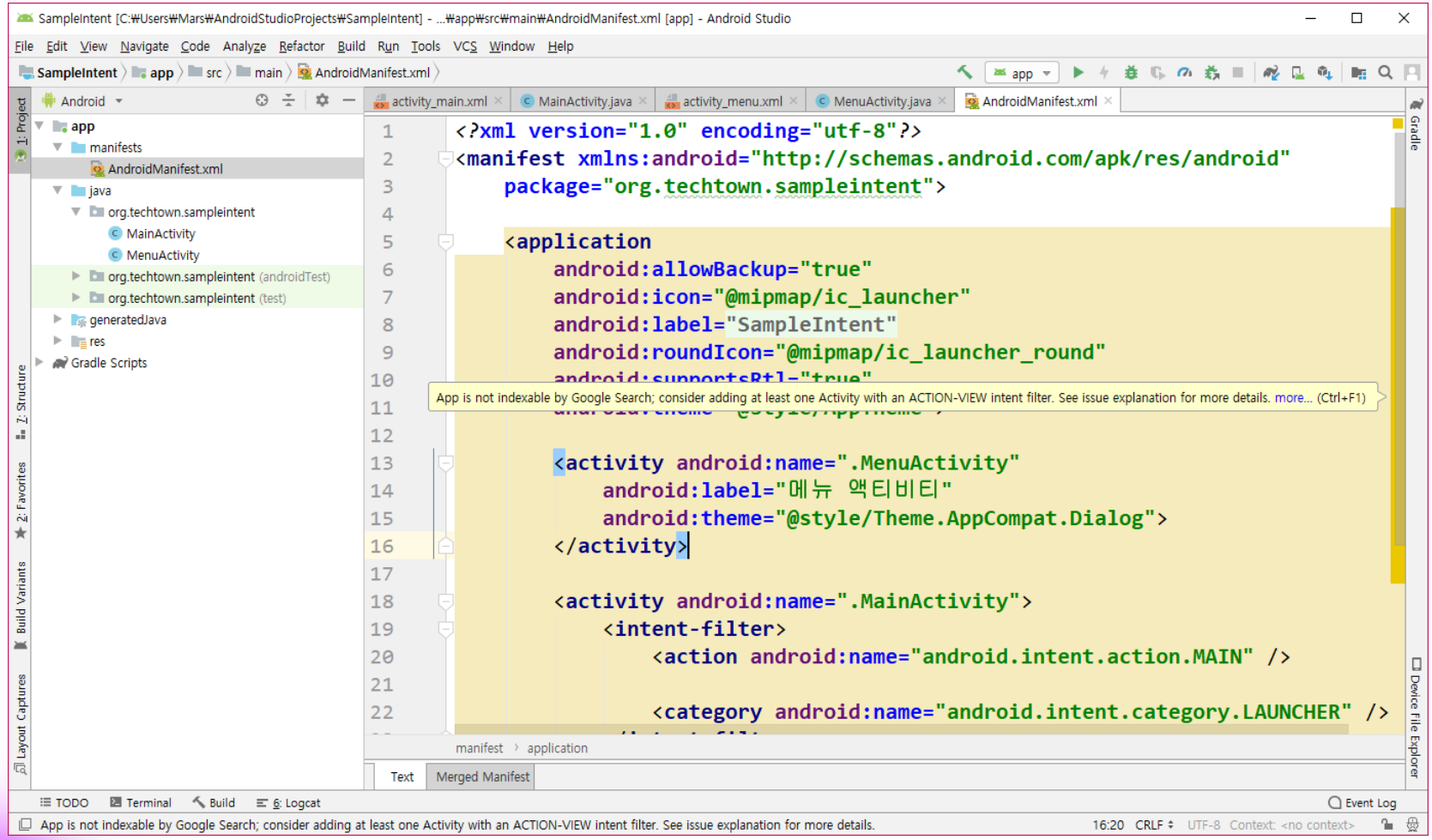
static View inflate (Context context, int resource, ViewGroup root)

2. 여러 화면 만들고 화면 간 전환하기

[안드로이드 애플리케이션을 구성하는 네 가지 구성요소]

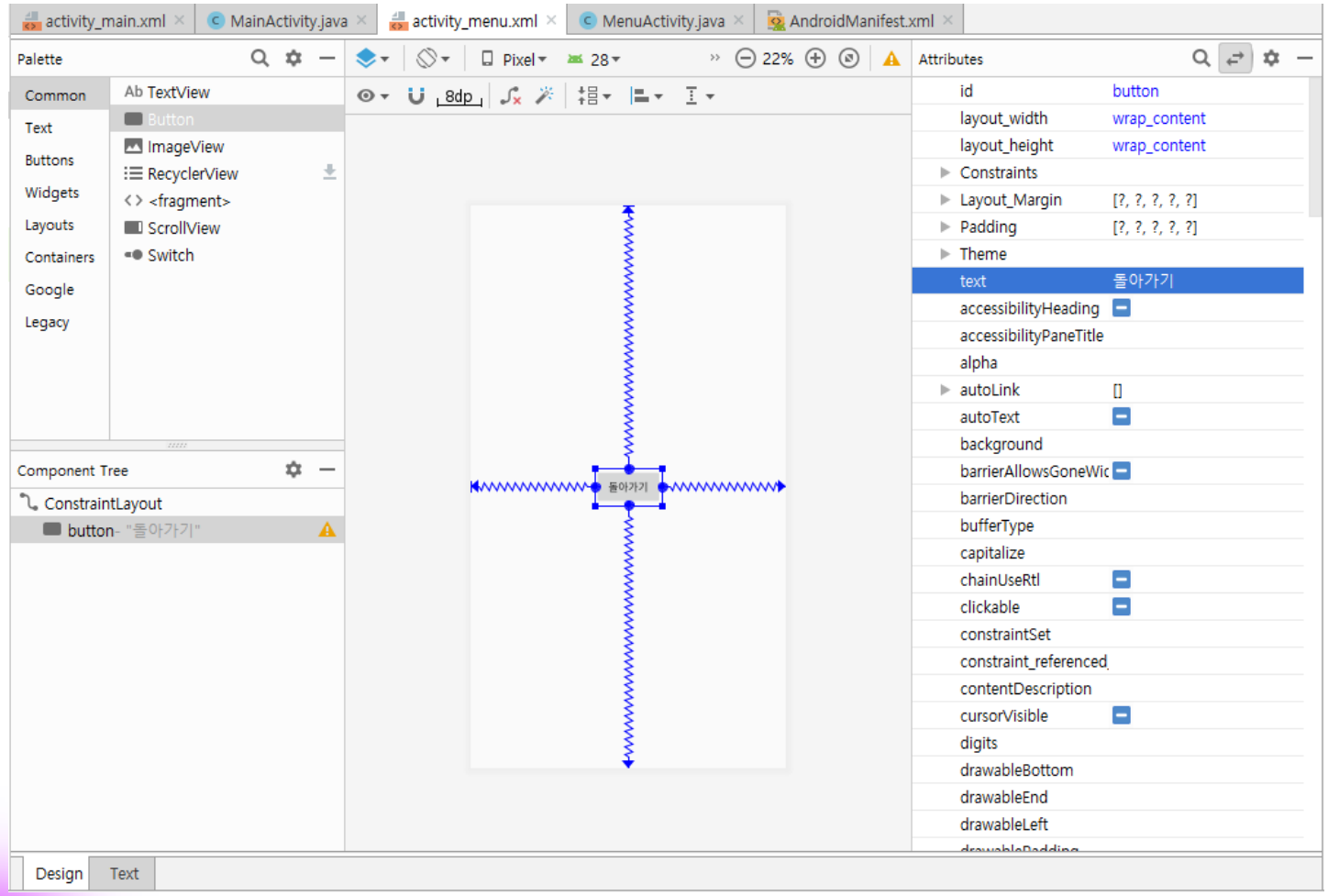
새로운 프로젝트 만들고 새 화면 추가

- SampleIntent 라는 이름으로 새로운 프로젝트 만들고 MenuActivity 추가



메뉴 화면에 버튼 추가

- activity_menu.xml 파일 열고 [돌아가기] 버튼 추가



돌아가기 버튼에 기능 추가

[Reference]

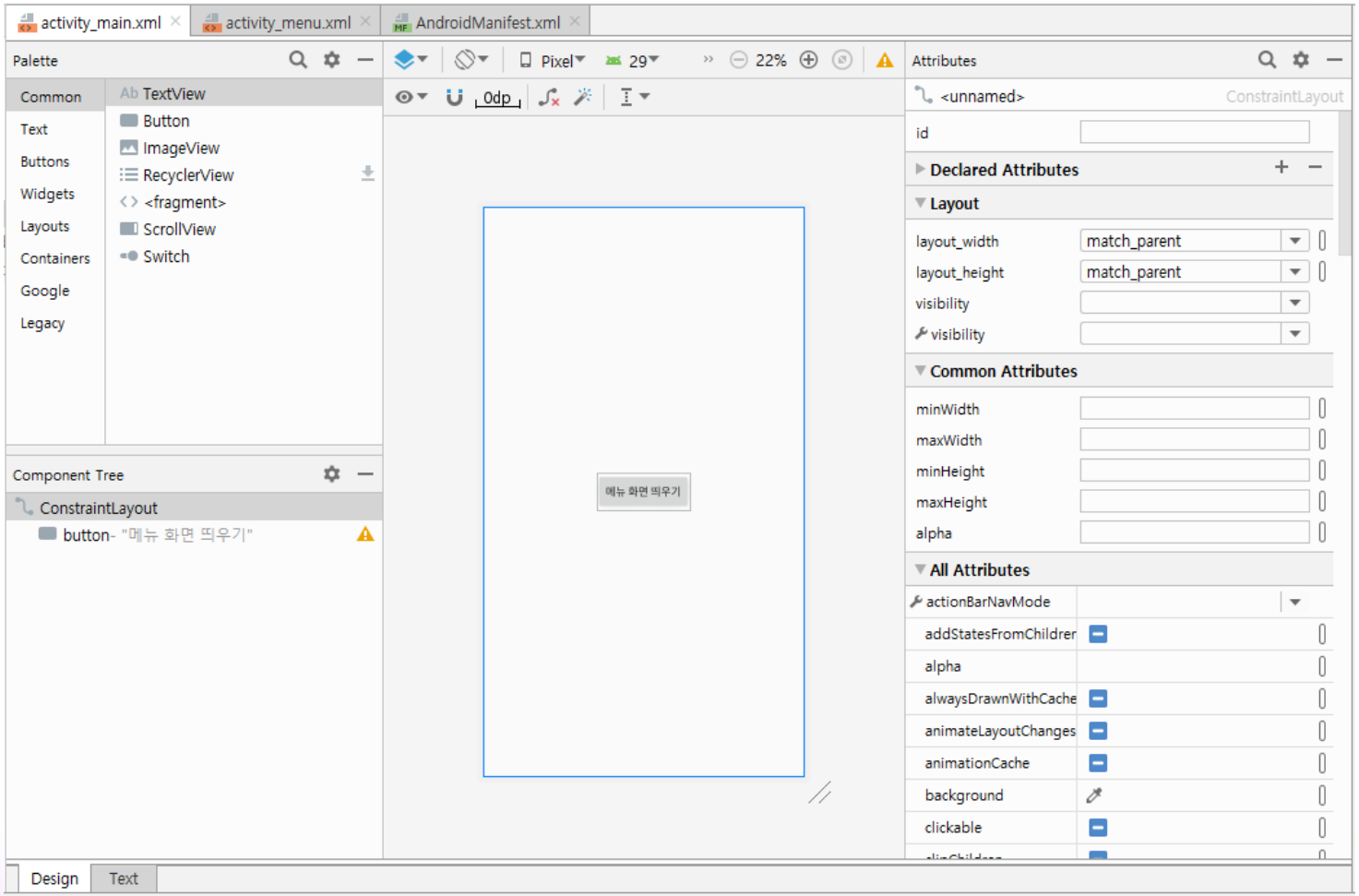
setResult(int resultCode, Intent intent)

참조파일 SampleIntent>/app/java/org.techtown.sampleintent/MenuActivity.java

```
중략…

public class MenuActivity extends AppCompatActivity {

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

        Button button = findViewById(R.id.button);  ❶ 버튼 객체 참조
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent();
                intent.putExtra("name", "mike");  ❷ 인텐트 객체 생성하고 name의 값을 부가 데이터로 넣기
                setResult(RESULT_OK, intent);  ❸ 응답 보내기
                finish();  ❹ 현재 액티비티 없애기
            }
        });
    }
}
```

메인 화면에 버튼 추가

- activity_main.xml 파일 열고 [메뉴화면 띄우기] 버튼 추가



메뉴화면 띄우기 버튼에 기능 추가

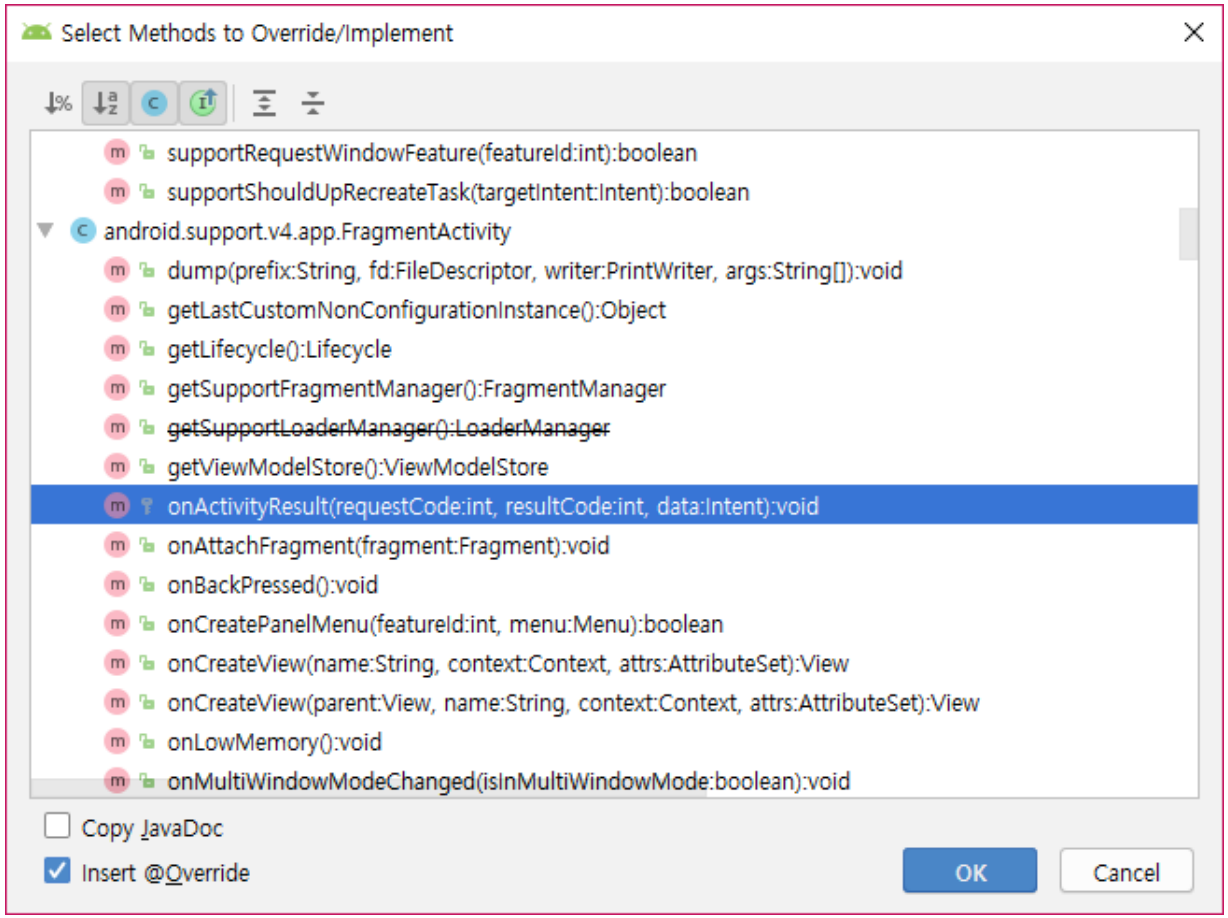
[Reference]

startActivityForResult(Intent intent, int requestCode)

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

응답을 받기 위한 메소드 추가

- onActivityResult 메소드 재정의

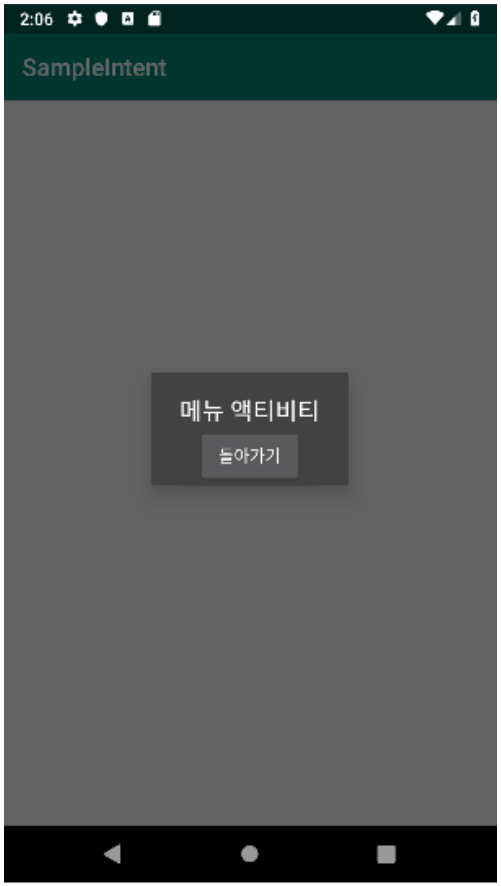


응답받은 결과를 토스트로 보여주기

```
protected void onActivityResult(int requestCode, int resultCode, 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 (resultCode == RESULT_OK) {
        String name = data.getExtras().getString("name");
        Toast.makeText(getApplicationContext(), "응답으로 전달된 name : " + name,
              Toast.LENGTH_LONG).show();
    }
  }
}
```

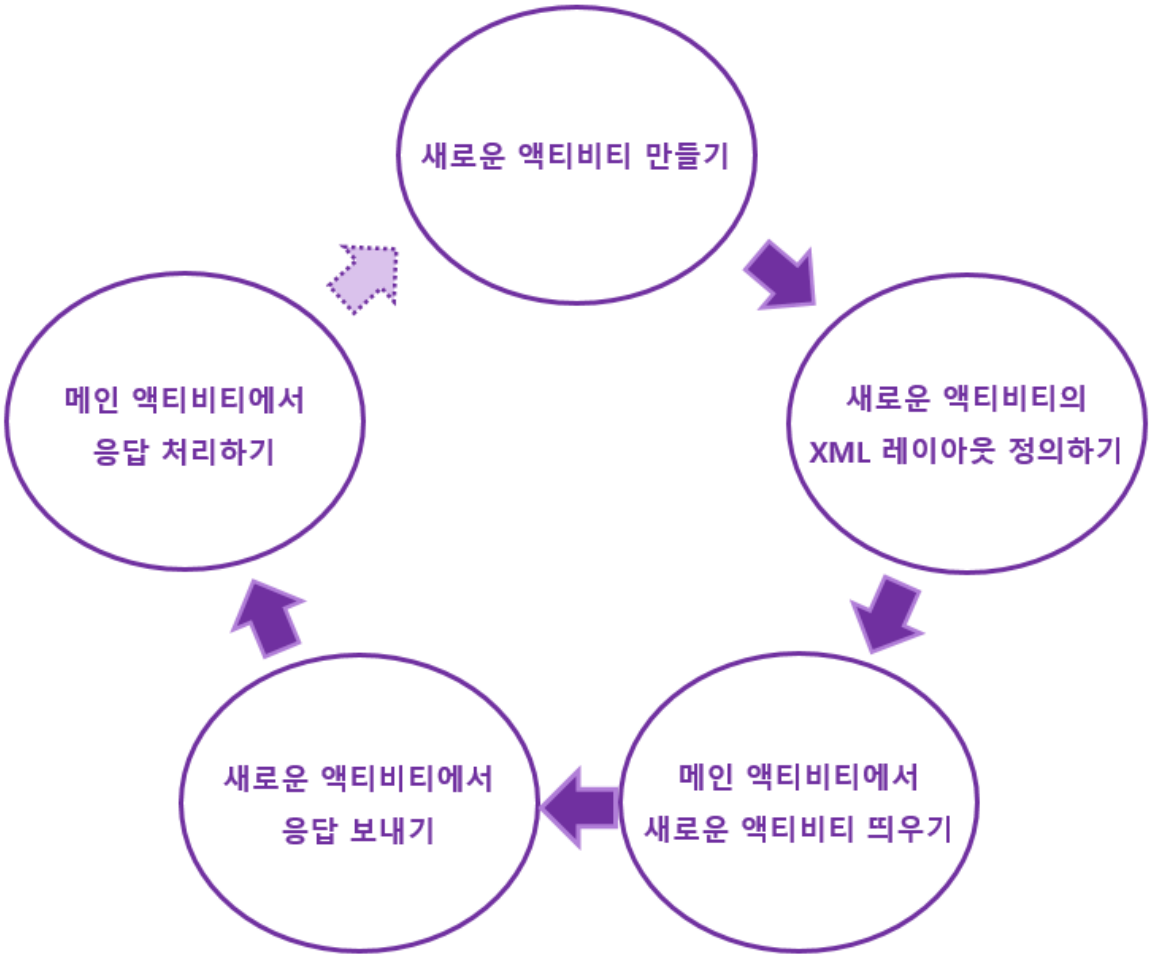
앱 실행 결과


2:01
SampleIntent
메뉴 화면 띄우기

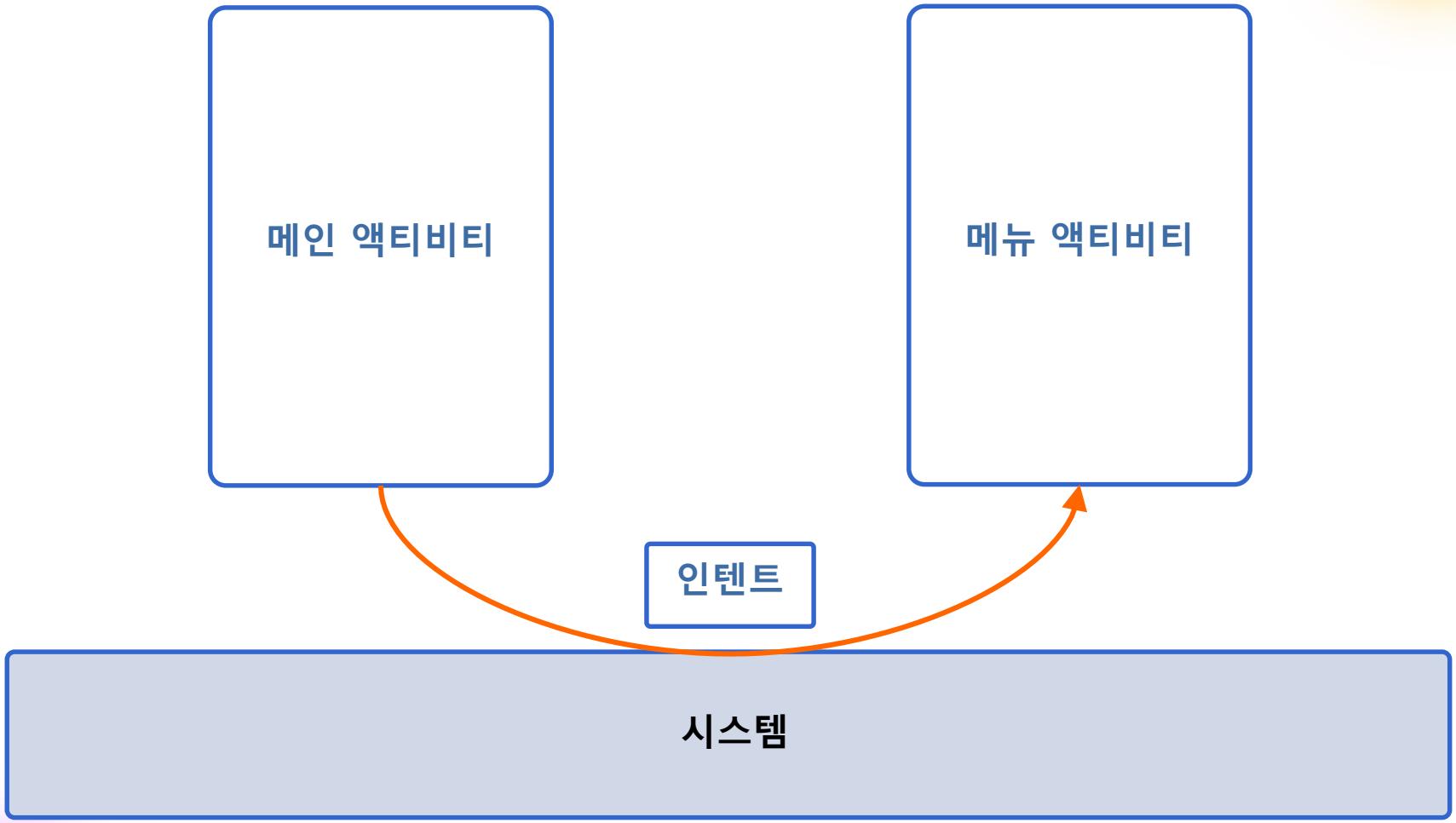

2:06
SampleIntent
메뉴 액티비티
돌아가기


2:07
SampleIntent
메뉴 화면 띄우기
응답으로 전달된 name : mike

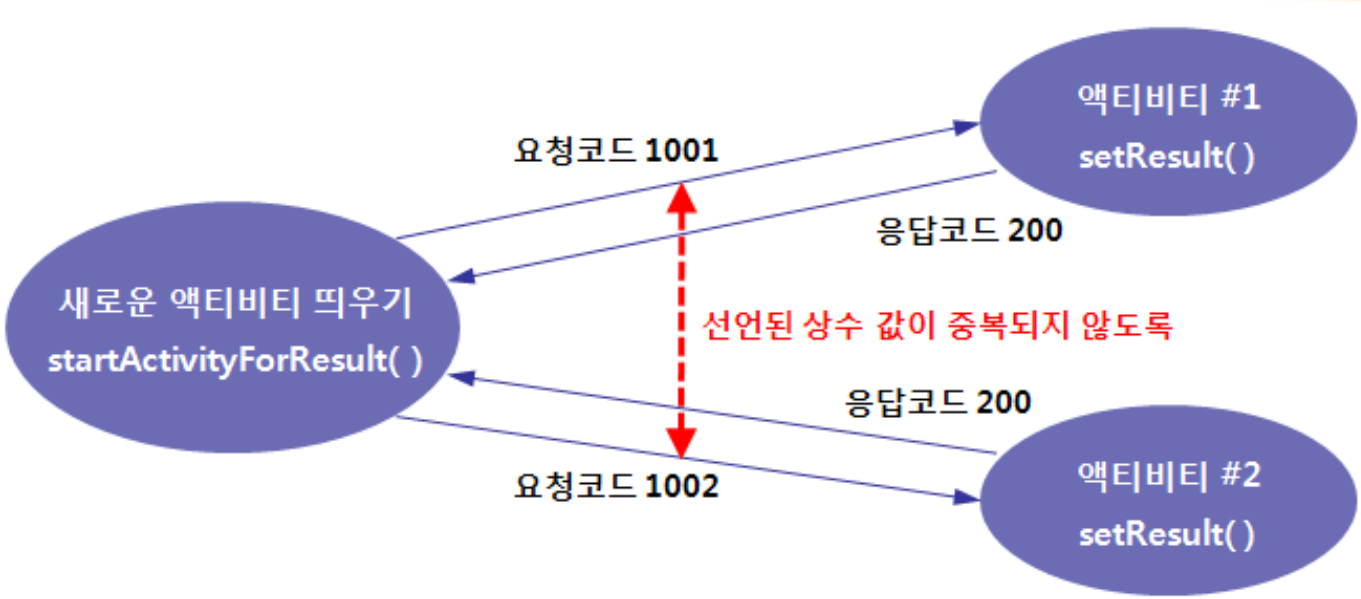
액티비티 구성 과정 정리


새로운 액티비티 만들기
새로운 액티비티의
XML 레이아웃 정의하기
메인 액티비티에서
새로운 액티비티 띄우기
새로운 액티비티에서
응답 보내기
메인 액티비티에서
응답 처리하기

화면을 띄울 때 시스템으로 요청하기



화면 구성과 화면 간 전환 과정



[액티비티에 선언된 상수 값 중복에 주의]

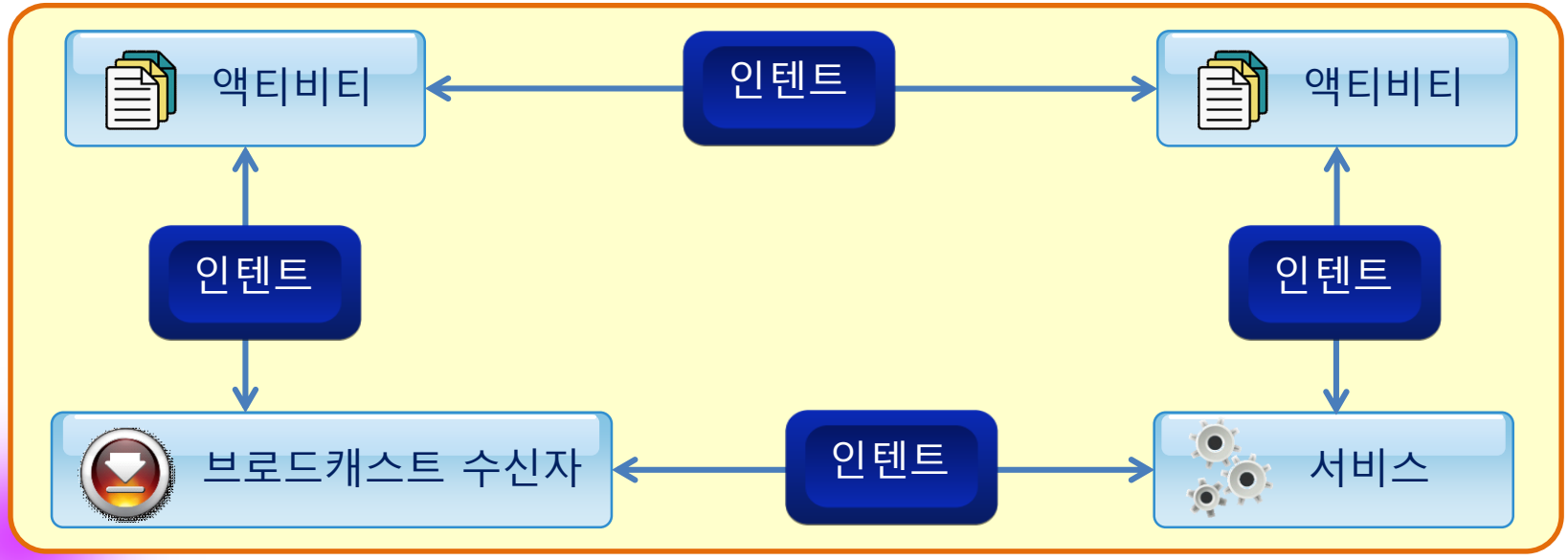
[Reference]

protected void onActivityResult(int requestCode, int resultCode, Intent Data)

3. 인텐트 살펴보기

인텐트와 데이터 전달

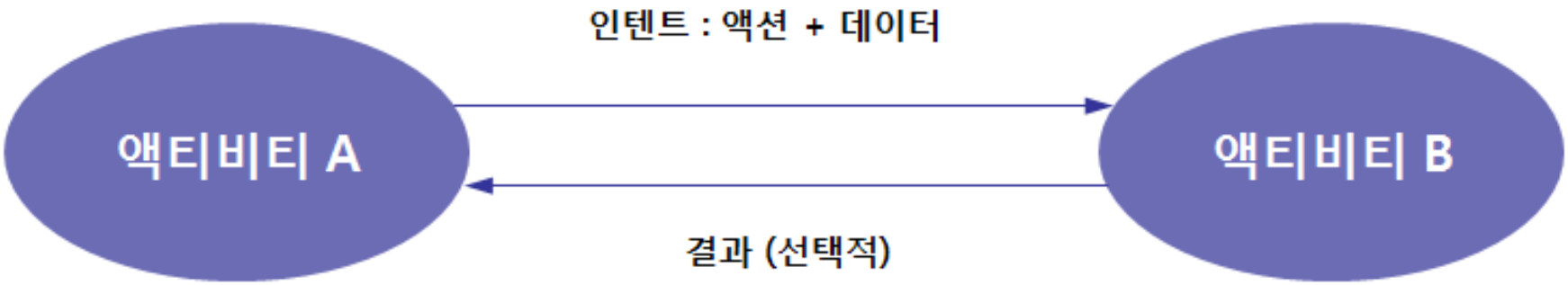

안드로이드 애플리케이션
액티비티
서비스
브로드캐스트 수신자


액티비티
인텐트
액티비티
인텐트
인텐트
브로드캐스트 수신자
인텐트
서비스

액티비티 간의 인텐트 전달

[Reference]

startActivity()
startService() 또는 bindService()
broadcastIntent()



[액티비티 간의 인텐트 전달]

액션과 데이터를 사용하는 대표적인 예

속성	설명
ACTION_DIAL tel:01077881234	주어진 전화번호를 이용해 전화걸기 화면을 보여줌
ACTION_VIEW tel:01077881234	주어진 전화번호를 이용해 전화걸기 화면을 보여줌. URI 값의 유형에 따라 VIEW 액션이 다른 기능을 수행함
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 인텐트와 암시적 인텐트

[Reference]

Intent()
Intent(Intent o)
Intent(String action [,Uri uri])
Intent(Context packageContext, Class<?> cls)
Intent(String action, Uri uri, Context packageContext, Class<?> cls)

- 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), 타입(Type), 컴포넌트(component), 부가 데이터(extras)

인텐트의 대표적 속성

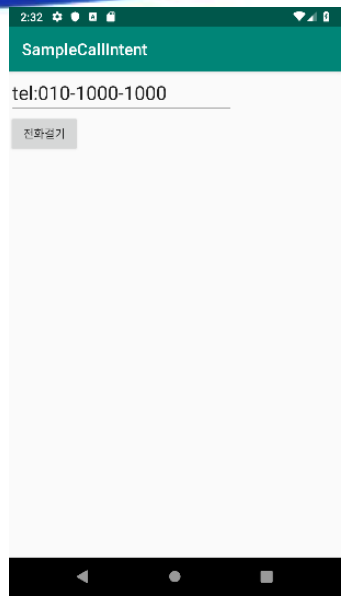
- **범주 (Category)**
  - 액션이 실행되는 데 필요한 추가적인 정보를 제공
- **타입 (Type)**
  - 인텐트에 들어가는 데이터의 MIME 타입을 명시적으로 지정
- **컴포넌트 (Component)**
  - 인텐트에 사용될 컴포넌트 클래스 이름을 명시적으로 지정
- **부가 데이터 (Extra)**
  - 인텐트는 추가적인 정보를 넣을 수 있도록 번들(Bundle) 객체를 담고 있음
  - 이 객체를 통해 인텐트 안에 더 많은 정보를 넣어 다른 애플리케이션 구성요소에 전달할 수 있음

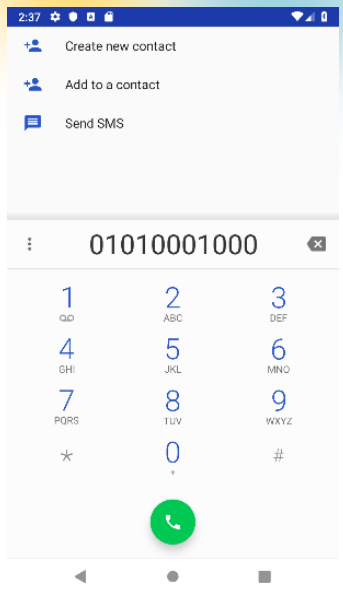
데이터 전달 예제

데이터 전달 예제

-액티비티 간에 인텐트로 데이터 전달

-전화걸기 기능 이용





XML 레이아웃 정의

- 전화걸기 화면을 띄우는 화면 구성

메인 액티비티 코드 작성

- 전화걸기 화면을 띄울 때 전화번호 전달

XML 레이아웃 만들기

참조파일 SampleCallIntent>/app/res/layout/activity_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"
    >

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="tel:010-1000-1000"
        android:textSize="24sp"
        />
```

❶ 전화번호를 입력할 입력상자 정의

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

</LinearLayout>
```

❷ 전화걸기 버튼 정의

메인 액티비티 코드 만들기

참조파일 SampleCallIntent>/app/java/org.techtown.samplecallintent/MainActivity.java

중략…

```
public class MainActivity extends AppCompatActivity {
    EditText editText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); ——> ❶ 뷰 객체 참조

        editText = findViewById(R.id.editText);

        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String data = editText.getText().toString(); ——> ❷ 입력상자에 입력된 전화번호 확인

                Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(data));——> ❸ 전화걸기 화면을 보여줄 인텐트 객체 생성
                startActivity(intent); ——> ❹ 액티비티 띄우기
            }
        });
    }
}
```

4. 플래그와 부가 데이터 사용하기

액티비티 스택


화면
새로운 액티비티 (new)
실행
실행 액티비티
정지/실행
재실행
액티비티 스택
이전 실행 액티비티
액티비티 #n-1
...
액티비티 #3
액티비티 #2
액티비티 #1

액티비티 스택과 플래그 사용

[Reference]

FLAG_ACTIVITY_SINGLE_TOP
FLAG_ACTIVITY_NO_HISTORY
FLAG_ACTIVITY_CLEAR_TOP

- 새로운 액티비티를 실행할 때마다 메모리에 새로운 객체를 만들고 이전 화면 위에 쌓는 방식은 비효율적일 수 있음
- 동일한 화면이 이미 만들어져 있는 경우에는 그 화면을 그대로 보여주고 싶다면 플래그를 사용하면 됨

[FLAG_ACTIVITY_SINGLE_TOP 플래그를 사용한 경우]

액티비티 플래그 사용 예

```
Intent intent = new Intent(getBaseContext(), AnotherActivity.class );
intent.putExtra("startCount", String.valueOf(startCount ));
intent.setFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP );
startActivityForResult(intent, REQUEST_CODE_ANOTHER );
```

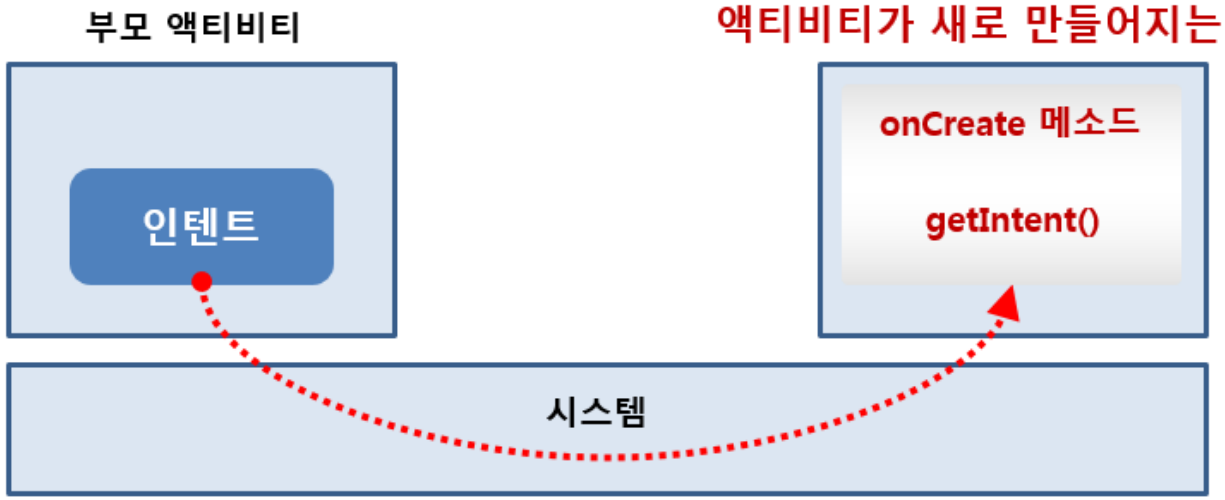
1 인텐트 객체 생성

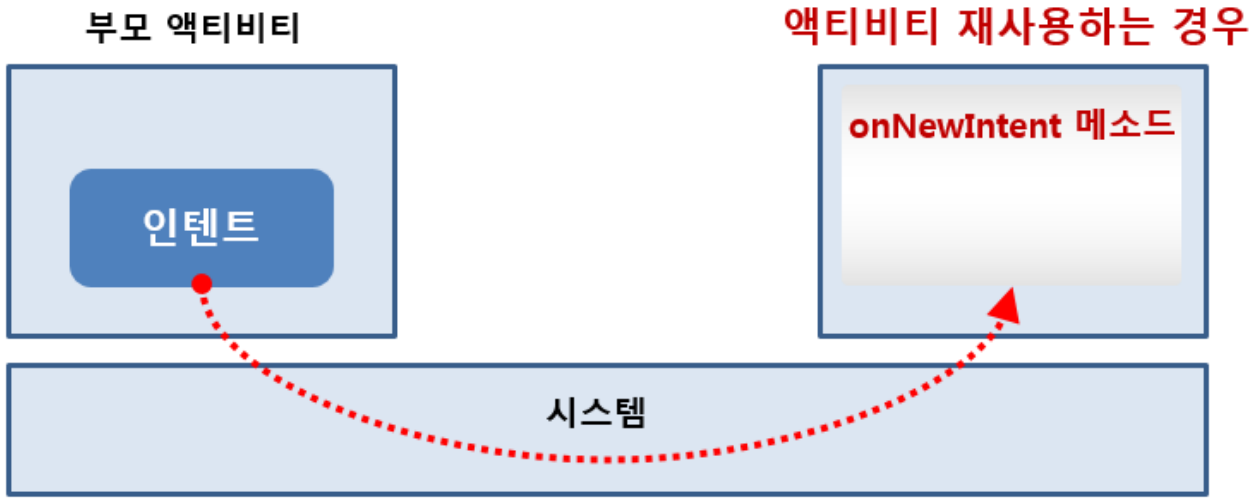
2 부가 데이터 넣기

3 인텐트 플래그 설정

4 인텐트 띄우기

액티비티에서 인텐트를 전달받는 두 가지 경우





다른 액티비티 플래그의 사용

[FLAG_ACTIVITY_NO_HISTORY 플래그를 사용한 경우]

[FLAG_ACTIVITY_CLEAR_TOP 플래그를 사용한 경우]

부가 데이터 전달하기

[Reference]

Intent putExtra(String name, String value)
Intent putExtra(String name, int value)
Intent putExtra(String name, boolean value)

String getStringExtra(String name)
int getIntExtra(String name, int defaultValue)
boolean getBooleanExtra(String name, boolean defaultValue)

[Reference]

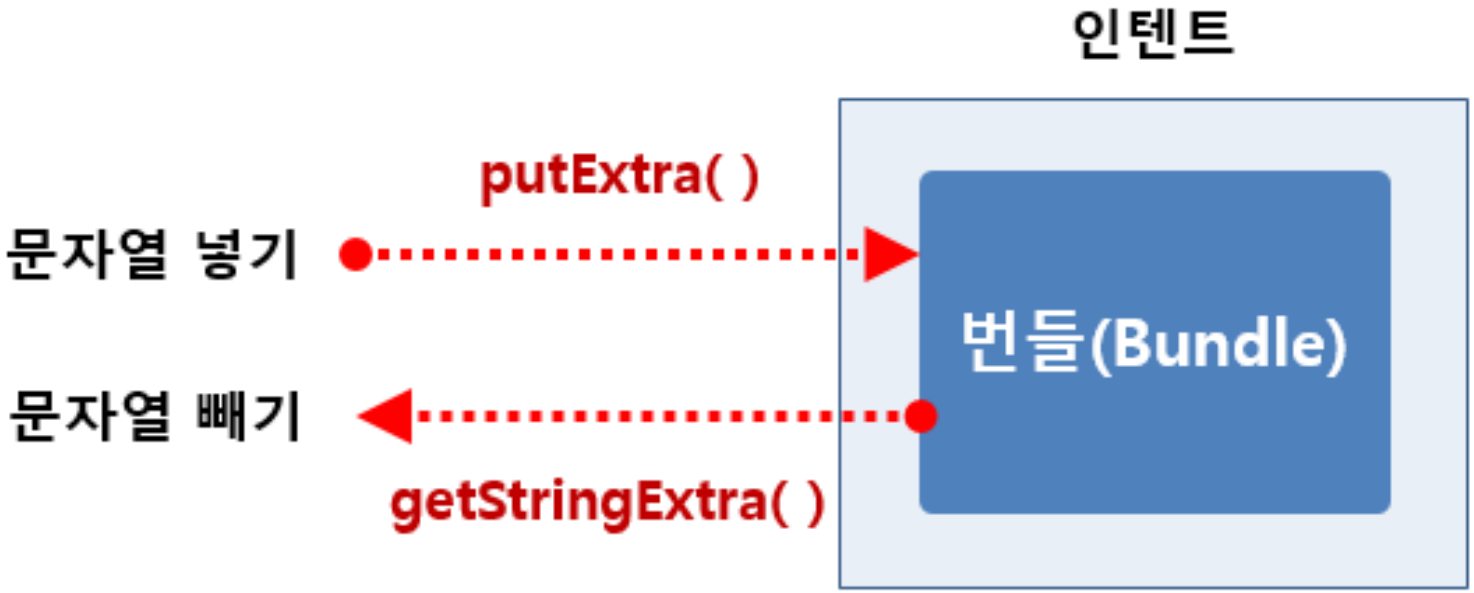
public abstract int describeContents()

public abstract void writeToParcel(Parcel dest, int flags)

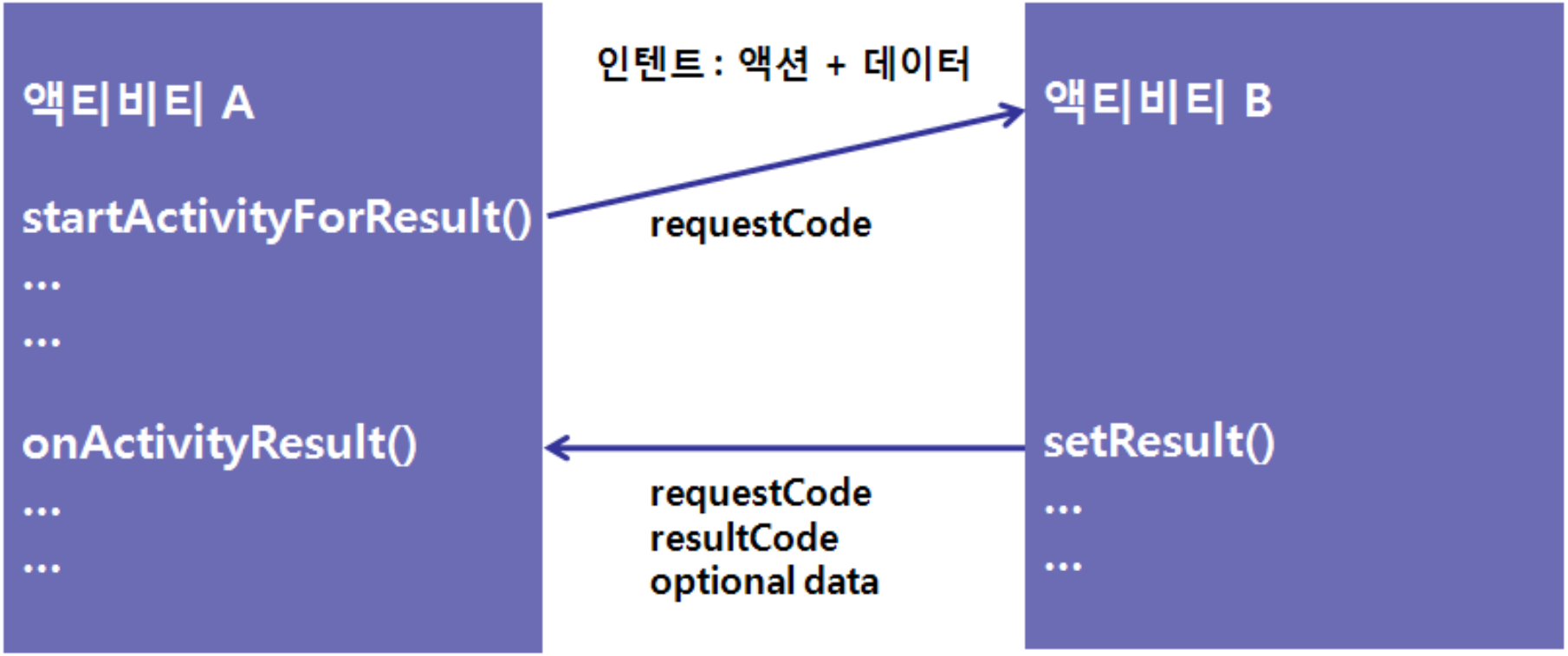
- 화면과 화면 간에 데이터를 전달하고 싶다면 인텐트의 부가 데이터(Extra)로 넣어 전달하는 방법을 사용함

- 인텐트는 애플리케이션 구성 요소 간에 데이터를 전달하는 방법을 제공하는 것이므로 화면과 화면 간 뿐만 아니라 화면과 서비스 간, 또는 브로드캐스트 수신자와 화면 간 등등 애플리케이션 구성 요소 간에 부가 데이터로 넣어 데이터를 전달할 수 있음

번들 객체

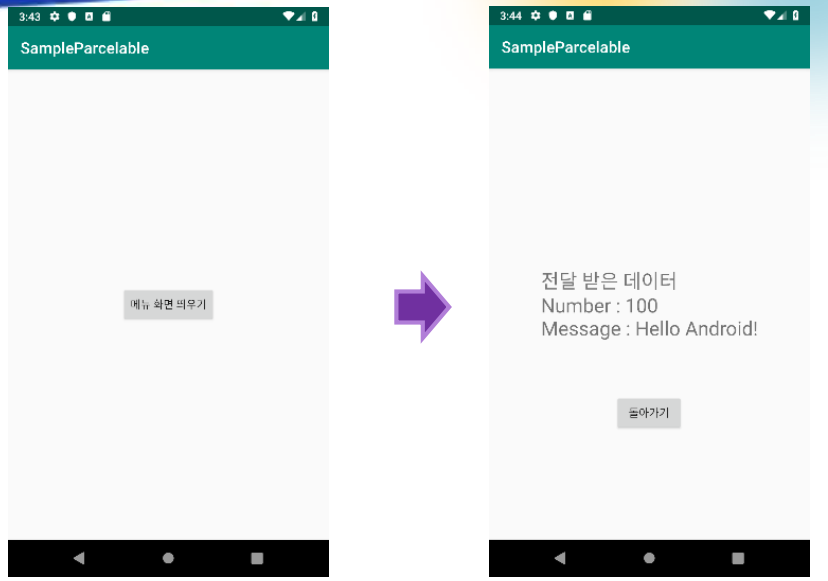


액티비티 간의 데이터 전달 방법 정리



Parcelable 예제

Parcelable 예제



-Parcelable 객체 만들기

-또다른 액티비티에 Parcelable 객체 전달하기

SimpleData 객체 정의

-Parcelable 인터페이스를 구현하는 객체 정의

메인 액티비티 코드 작성

-또 다른 액티비티를 띄워줄 때 부가데이터 전달

또다른 액티비티 코드 작성

-전달된 객체를 이용해 데이터 표시

SimpleData 클래스 정의

```
public class SimpleData implements Parcelable {
 int number;
 String message;
 public SimpleData(int num, String msg) {
  number = num;
  message = msg;
 }

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

 public static final Parcelable.Creator CREATOR = new Parcelable.Creator() {
   public SimpleData createFromParcel(Parcel in) {
    return new SimpleData (in);
   }
 public SimpleData[] newArray(int size) {
  return new SimpleData[size];
 }
};
```

Continued..

SimpleData 클래스 정의 (계속)

```
  public int describeContents() {
    return 0;
  }
  public void writeToParcel(Parcel dest, int flags) {
    dest.writeInt(number);
    dest.writeString(message);
  }
  public int getNumber() {
    return number;
  }
  public void setNumber(int number) {
    this.number = number;
  }
  public String getMessage() {
    return message;
  }
  public void setMessage(String message) {
    this.message = message;
  }
}
```

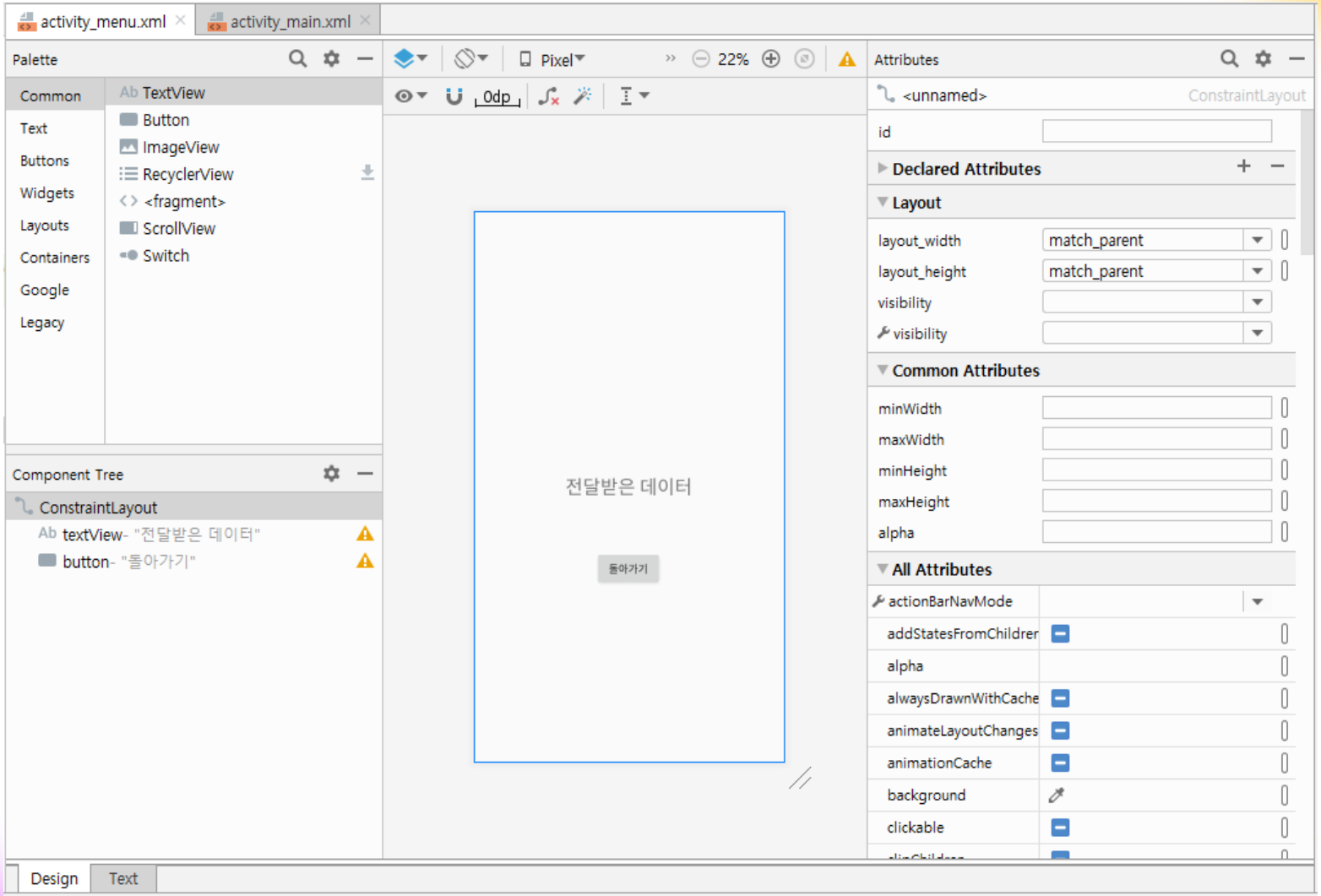
메인 액티비티 코드 만들기

```
...
public static final String KEY_SIMPLE_DATA = "data";
...

    public void onButton1Clicked(View v) {
     Intent intent = new Intent(getApplicationContext(), AnotherActivity.class);
     SimpleData data = new SimpleData(100, "Hello Android!" );
     intent.putExtra(KEY_SIMPLE_DATA, data );
     startActivity(intent);
     }
```

1 SimpleData 객체 생성

2 인텐트에 부가 데이터로 넣기

메뉴 액티비티의 화면 레이아웃



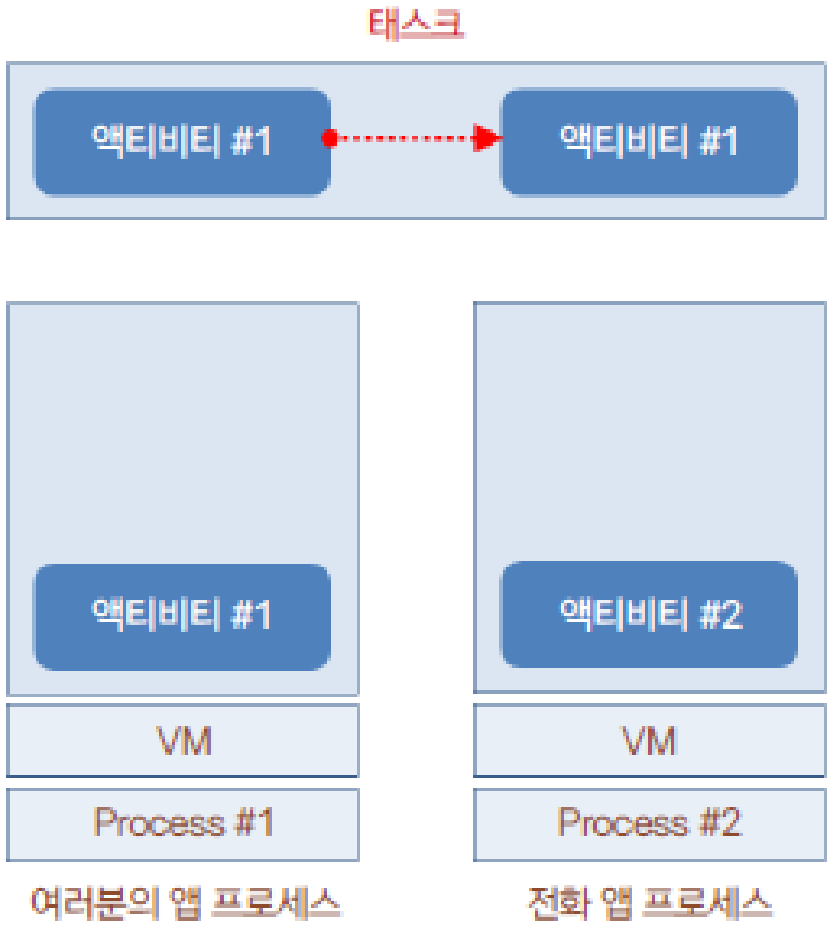
메뉴 액티비티 코드 만들기

```
    ...
    Intent intent = getIntent();
    processIntent(intent);
}

private void processIntent(Intent intent) {
    if (intent != null) {
        Bundle bundle = intent.getExtras();
        SimpleData data = (SimpleData) bundle.getParcelable(KEY_SIMPLE_DATA);

        textView.setText("전달 받은 데이터₩nNumber : " + data.getNumber()
                        + "₩nMessage : " + data.getMessage());
    }
}
```

5. 태스크 관리 이해하기

태스크의 역할

- 태스크는 프로세스와 관계없이 화면의 흐름 관리

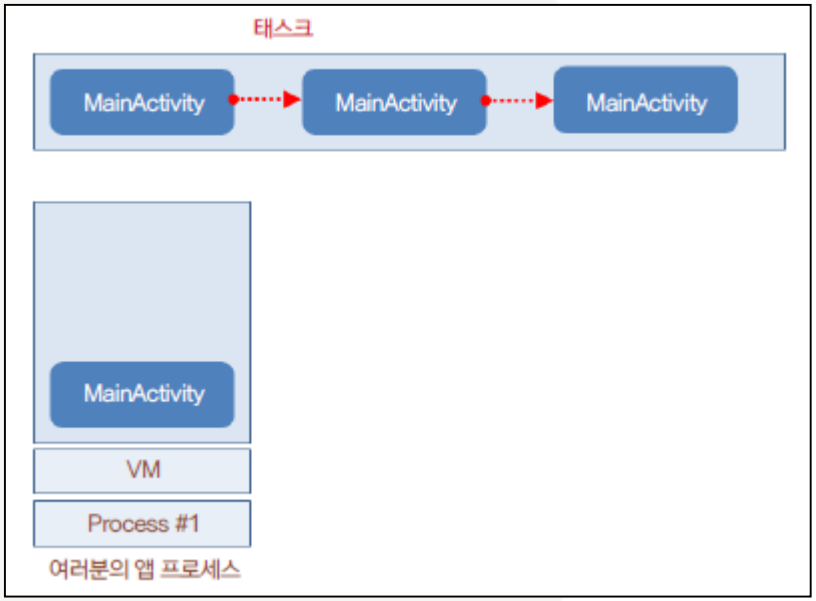


태스크의 역할

- 메인 액티비티를 여러 번 띄울 때의 태스크

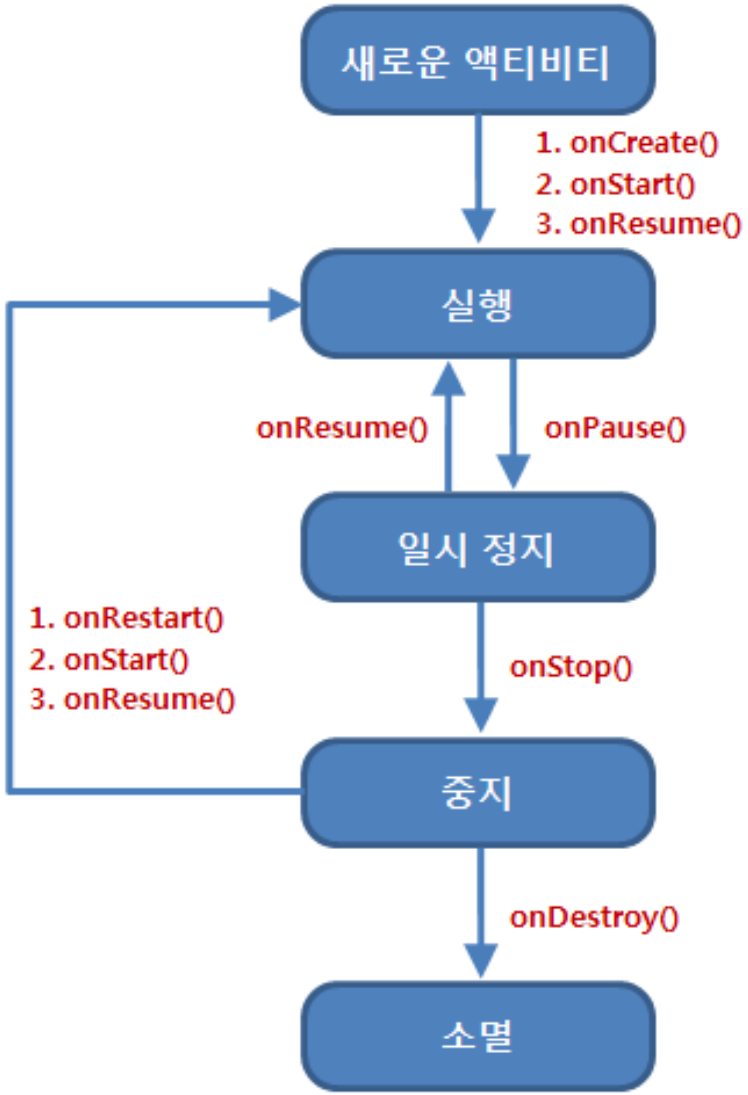
참조파일 SampleTask>/app/src/org.techtown.sampletask/MainActivity.java

```
public class MainActivity extends AppCompatActivity {

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

    Button button = findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(intent);
      }
    });
  }
}
```



6. 액티비티의 수명주기와 SharedPreferences 이해하기

수명주기에 따른 상태 변화



액티비티의 상태 메소드

상태 메소드	설 명
onCreate()	- 액티비티가 처음에 만들어졌을 때 호출됨 - 화면에 보이는 뷰들의 일반적인 상태를 설정하는 부분 - 이전 상태가 저장되어 있는 경우에는 번들 객체를 참조하여 이전 상태 복원 가능 - 이 메소드 다음에는 항상 onStart() 메소드가 호출됨
onStart()	- 액티비티가 화면에 보이기 바로 전에 호출됨 - 액티비티가 화면 상에 보이면 이 메소드 다음에 onResume() 메소드가 호출됨 - 액티비티가 화면에서 가려지게 되면 이 메소드 다음에 onStop() 메소드가 호출됨
onResume()	- 액티비티가 사용자와 상호작용하기 바로 전에 호출됨
onRestart()	- 액티비티가 중지된 이후에 호출되는 메소드로 다시 시작되기 바로 전에 호출됨 - 이 메소드 다음에는 항상 onStart() 메소드가 호출됨
onPause()	- 또 다른 액티비티를 시작하려고 할 때 호출됨 - 저장되지 않은 데이터를 저장소에 저장하거나 애니메이션 중인 작업을 중지하는 등의 기능을 수행하는 메소드임 - 이 메소드가 리턴하기 전에는 다음 액티비티가 시작될 수 없으므로 이 작업은 매우 빨리 수행된 후 리턴되어야 함 - 액티비티가 이 상태에 들어가면 시스템은 액티비티를 강제 종료할 수 있음
onStop()	- 액티비티가 사용자에게 더 이상 보이지 않을 때 호출됨 - 액티비티가 소멸되거나 또 다른 액티비티가 화면을 가릴 때 호출됨 - 액티비티가 이 상태에 들어가면 시스템은 액티비티를 강제 종료할 수 있음
onDestroy()	- 액티비티가 소멸되어 없어지기 전에 호출됨 - 이 메소드는 액티비티가 받는 마지막 호출이 됨 - 액티비티가 애플리케이션에 의해 종료되거나(finish() 메소드 호출) 시스템이 강제로 종료시키는 경우에 호출될 수 있음 - 위의 두 가지 경우를 구분할 때 isFinishing() 메소드를 이용함 - 액티비티가 이 상태에 들어가면 시스템은 액티비티를 강제 종료할 수 있음

수명주기 확인하기 예제

수명주기 확인하기 예제

-액티비티 상태에 따른 수명주기 확인하기

-상태 메소드 별로 토스트 메시지 추가

XML 레이아웃 정의

-입력상자와 버튼이 있는 레이아웃

메인 액티비티 코드 작성

-상태 메소드 별로 토스트 메시지 코드 추가

메인 액티비티 코드 만들기

```
 @Override
protected void onDestroy() {
 super.onDestroy();
 Toast.makeText(getBaseContext(), "onDestroy ...", Toast.LENGTH_LONG).show();
}
@Override
protected void onPause() {
 super.onPause();
 saveState();
 Toast.makeText(getBaseContext(), "onPause ...", Toast.LENGTH_LONG).show();
}
@Override
protected void onRestart() {
 super.onRestart();
 Toast.makeText(getBaseContext(), "onRestart ...", Toast.LENGTH_LONG).show();
}
```

현재 상태 저장

Continued..

메인 액티비티 코드 만들기 (계속)

```
@Override
protected void onResume() {
  super.onResume();
  restoreState();
  Toast.makeText(getBaseContext(), "onResume...", Toast.LENGTH_LONG).show();
}
@Override
protected void onStart() {
  super.onStart();
  Toast.makeText(getBaseContext(), "onStart ...", Toast.LENGTH_LONG).show();
}
@Override
protected void onStop() {
  super.onStop();
  Toast.makeText(getBaseContext(), "onStop ...", Toast.LENGTH_LONG).show();
}
```

현재 상태 복원

Continued..

메인 액티비티 코드 만들기 (계속)

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