

Android 애플리케이션 프로그래밍 – Activity

Activity, Intent, 서브.양방향 Activity



부산대학교 정보·의생명공학대학
정보컴퓨터공학부



이론

강의 목표와 구성

❖ Activity

- Component 란?
- Intent 이해하기
- Activity와 라이프 사이클
- Intent를 통한 양방향 Activity

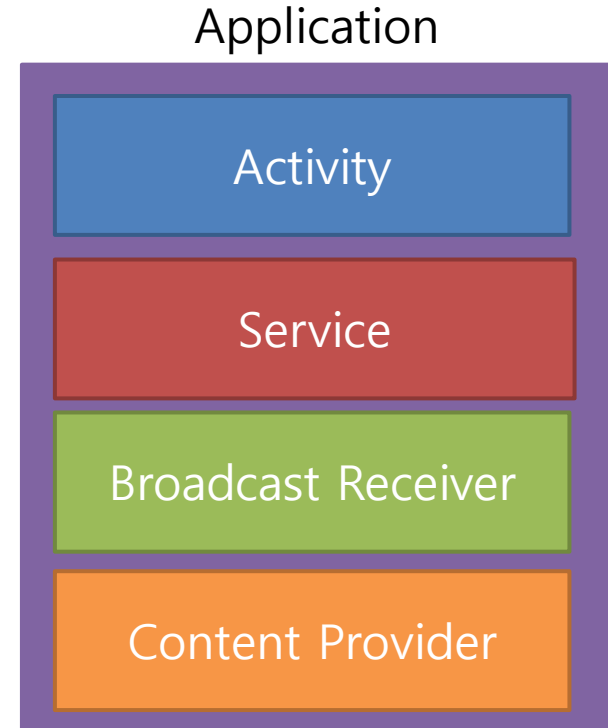
Component

❖ Component 란?

- Android 아키텍처의 가장 중요한 요소
- 컴포넌트는 앱의 구성 단위이며, 컴포넌트 여러 개를 조합하여 하나의 앱을 만듦
- 컴포넌트는 앱 내에서 독립적인 실행 단위임
- main 함수 같은 애플리케이션의 진입 지점이 따로 없음

❖ Component 종류와 특징

- **Activity:** 화면을 구성하는 가장 기본적인 Component
- **Service:** Activity와 상관없이 백그라운드에서 동작하는 Component
- **Broadcast Receiver:** 응용프로그램 및 장치에 메시지를 전달하기 위해 사용되는 Broadcasting 메시지가 발생하면 반응하는 Component
- **Content Provider:** 제공자, 응용프로그램 사이에 데이터를 공유하기 위한 Component

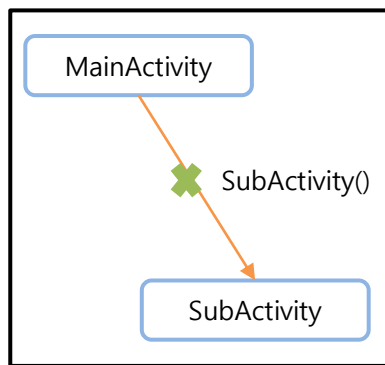


Intent 이해하기

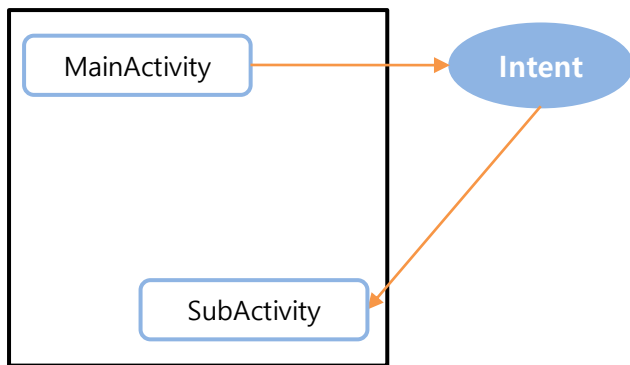
❖ Intent 란?

- **Component를 실행**하기 위해 **시스템에 전달하는 메시지**
- 기능을 수행하는 함수를 제공하는 것이 아닌 **데이터를 담은 클래스**
 - 데이터는 Component를 실행하는 정보이며, 정보가 담긴 Intent 객체를 시스템에 전달하면 Component가 실행된다.

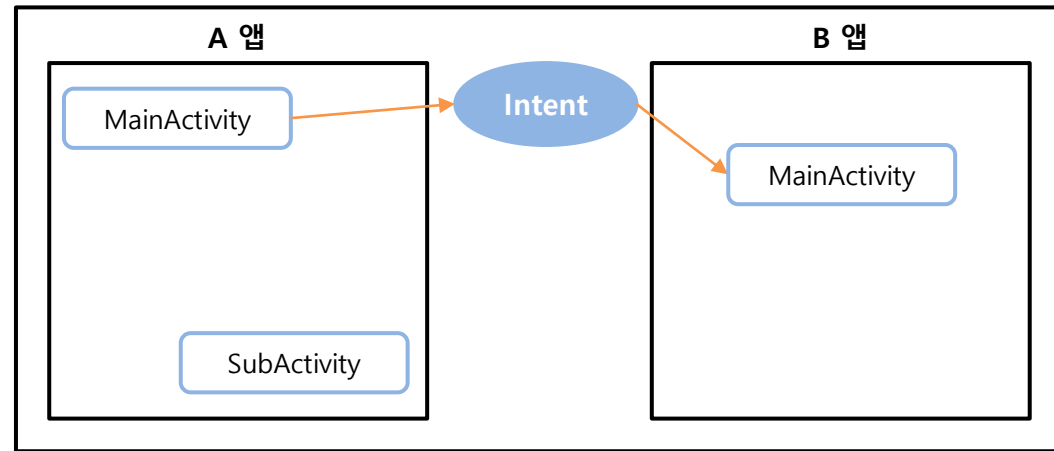
❖ Component와 Intent



Component 직접 실행



Intent로 Component 실행



Intent로 외부 앱 연동

- **Android Component 클래스**는 **시스템이 생성하고 실행하는 클래스**이므로 직접 생성할 수 없다.
- 시스템에 Intent를 통해 실행할 Component에 대한 정보를 전달하여 타겟 Component를 실행한다.
- Intent를 통해 외부 앱의 Component를 실행할 수 있다.

Intent 이해하기

❖ Intent 예시

- 버튼 클릭 시 SubActivity 클래스를 포함하는 Intent 시작

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val btn : Button = findViewById(R.id.btn)           버튼 객체 선언  
        btn.setOnClickListener { it: View!                  Intent 객체 선언  
            val intent = Intent(packageContext: this, SubActivity::class.java)  
            startActivity(intent)  
        }  
    }  
}
```

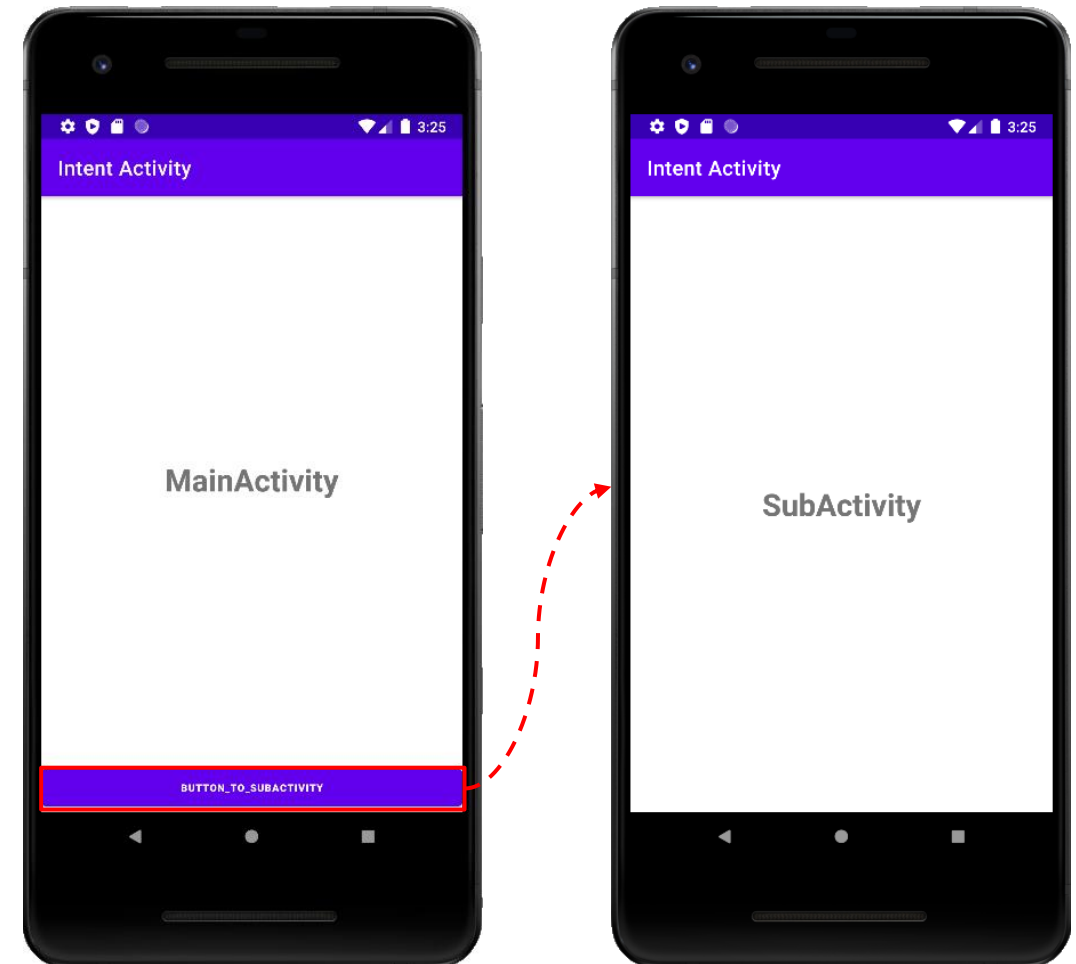


그림 버튼을 이용한 Intent로 SubActivity 실행

Activity와 라이프 사이클

❖ Activity 란?

- Android 앱에서 **화면을 구성하는 Component**
- 앱 실행 시 지정된 Activity를 실행하여 사용자에게 표시함

❖ Activity 의 특징

- Activity는 **매니페스트 파일에 등록 필요**
- **Activity 하나당 <activity> 태그 하나로 등록 필요**
 - Activity 뿐만 아니라 service, broadcast receiver, content provider 도 매니페스트 파일에 등록해야 한다.

```
<activity android:name=".SubActivity"
/>
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Manifest 파일의 Activity 선언

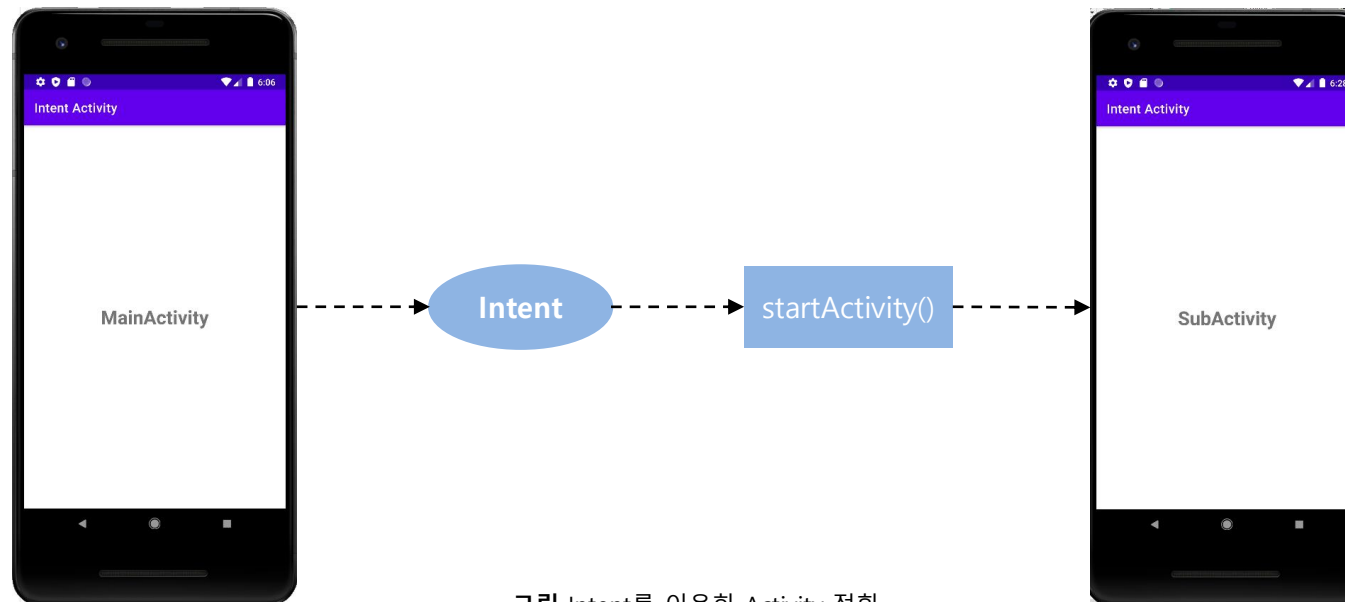


그림 Intent를 이용한 Activity 전환

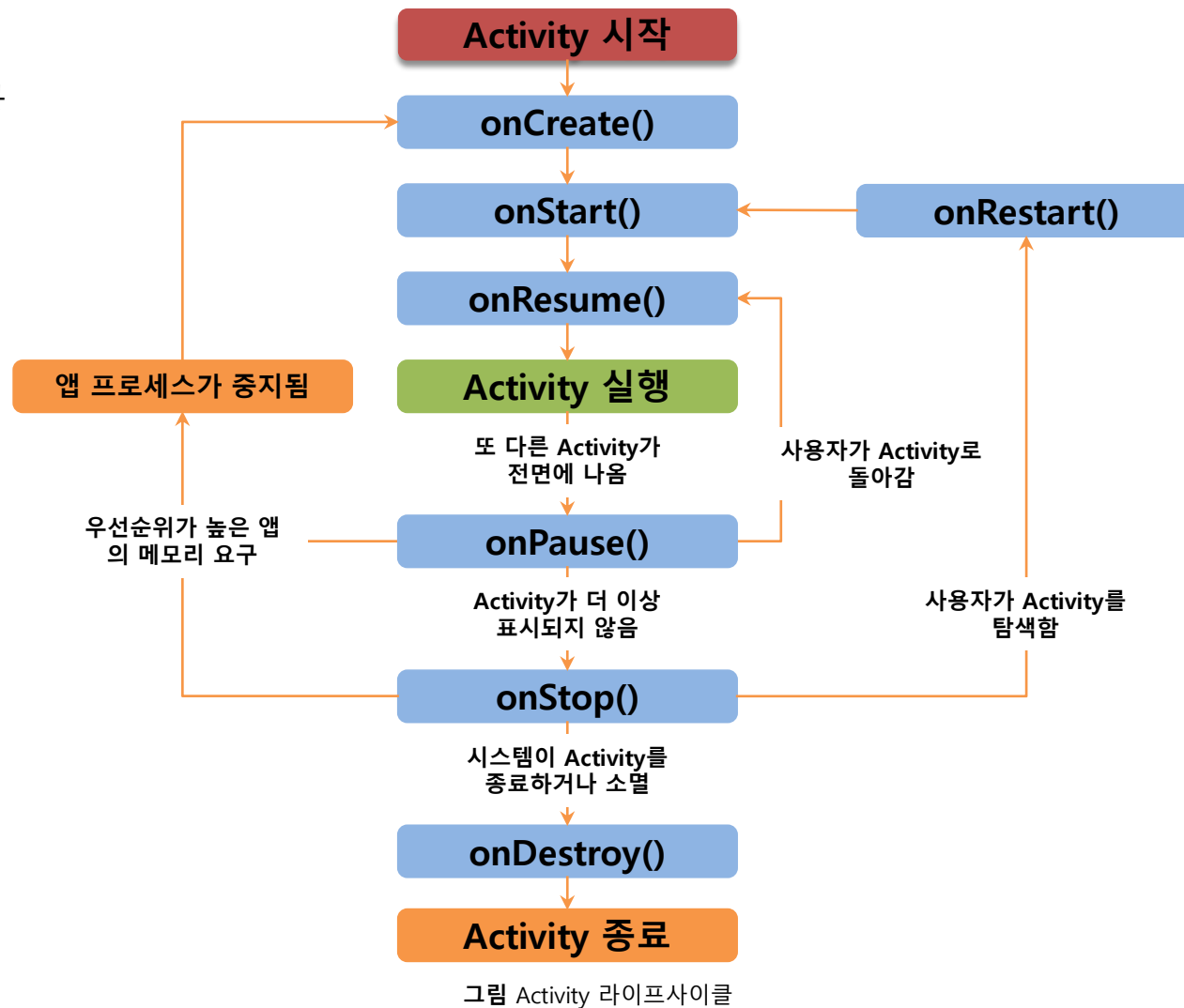
Activity와 라이프 사이클

❖ Activity 라이프사이클

- Activity가 생성되어 소멸하기까지의 과정
- Activity 클래스는 Activity의 상태 변화 관련 메소드 제공

❖ Activity의 상태

- **활성** : Activity 화면이 출력되고 있고 사용자가 이벤트를 발생시킬 수 있는 상태
- **일시 정지** : Activity의 화면이 출력되고 있지만 사용자가 이벤트를 발생시킬 수 없는 상태
- **비활성** : Activity의 화면이 출력되고 있지 않은 상태



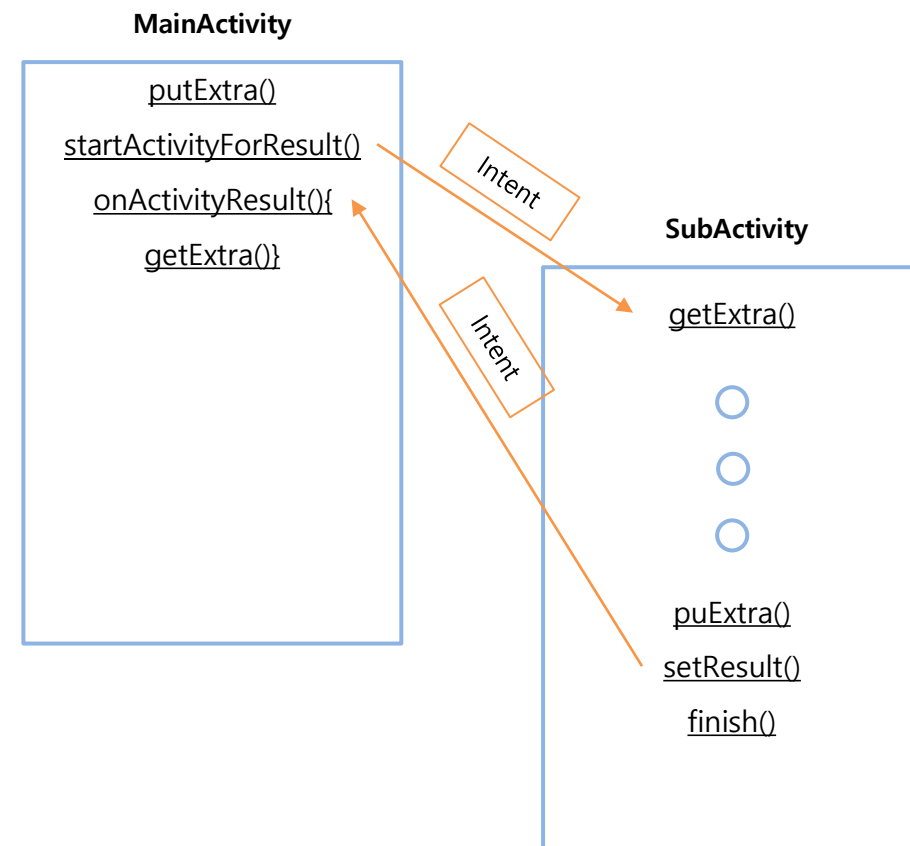
Intent를 통한 양방향 Activity

❖ 양방향 Activity란?

- MainActivity에서 서브 Activity로 데이터를 넘긴 후 SubActivity에서 다시 MainActivity로 데이터를 돌려주는 과정

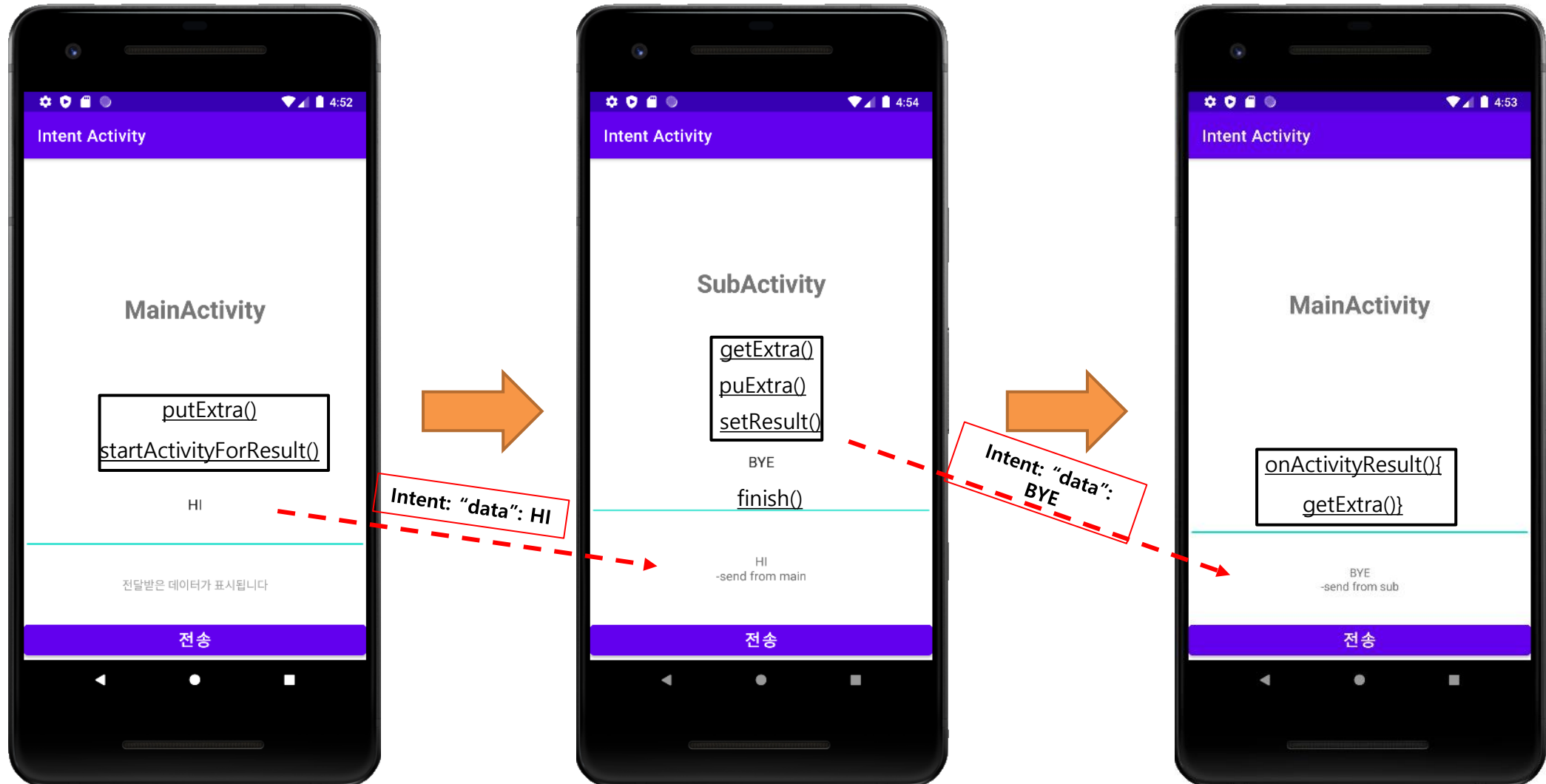
❖ 양방향 Activity 진행 과정

- MainActivity에서 Intent에 putExtra()로 보낼 데이터를 넣는다.
- SubActivity로부터 결과값을 돌려받기 위해 startActivityForResult()메소드를 사용해 Intent를 전송한다.
- SubActivity에서 getExtra() 메소드를 사용하여 데이터를 받는다.
- SubActivity에서 받은 데이터를 처리과정을 진행한 후 리턴 할 데이터를 putExtra() 메소드로 Intent에 보낼 데이터를 저장합니다.
- setResult()메소드를 통해 메인 Activity로 데이터를 리턴한다.
- SubActivity는 finish() 메소드를 호출하여 종료된다.
- MainActivity에서는 onActivityResult() 메소드를 오버라이딩 하고, 오버라이딩 된 메소드 안에서 getExtra() 메소드를 호출하여 데이터를 돌려받는다.



Intent를 통한 양방향 Activity

❖ 양방향 Activity 실행 결과



실습

실습 목표와 구성

1. 기초(따라하기)

- Intent를 통한 액티비티 전환

2. 응용(로직구현)

- 양방향 액티비티

3. 심화(완성하기)

- 연락처 만들기

4. 심화(과제물)

- 자기소개서 만들기

기초(따라하기) – 예제 1

❖ Intent를 통한 액티비티 전환

1. 메인 Activity 레이아웃 구성
2. activity_sub.xml 생성
3. 서브 Activity 레이아웃 구성
4. SubActivity.kt 생성
5. 서브 액티비티 구현
6. Intent, 메인 액티비티 구현
7. 새로운 액티비티 등록(AndroidManifest.xml 수정)



기초(따라하기) – 예제 1

1. 메인 액티비티 레이아웃 구성



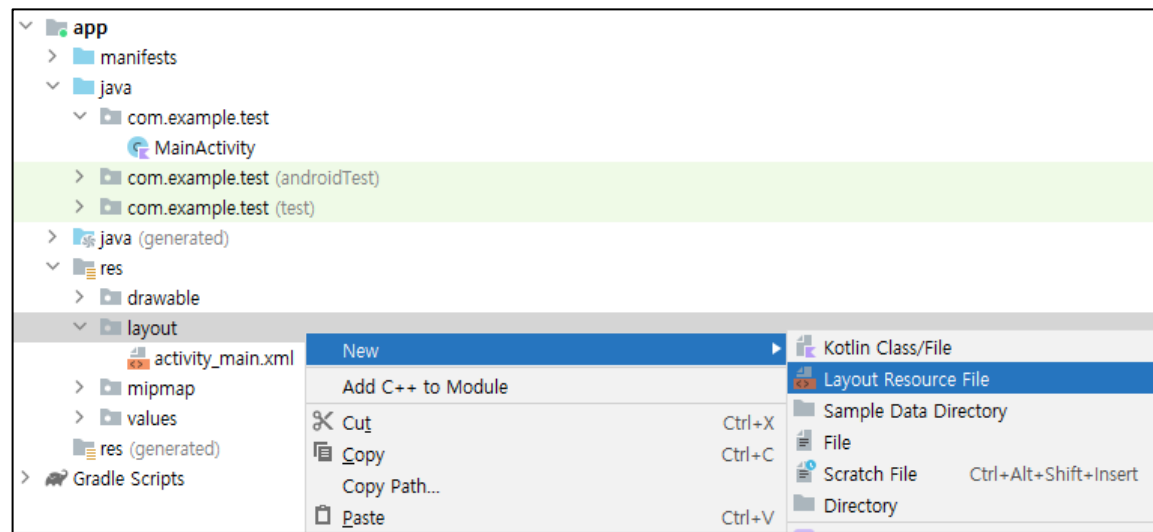
activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6      <TextView
7          android:layout_width="match_parent"
8          android:layout_height="wrap_content"
9          android:layout_weight="5"
10         android:gravity="center"
11         android:text="MainActivity"
12         android:textSize="30dp"
13         android:textStyle="bold" />
14
15     <Button
16         android:id="@+id/btn_main"
17         android:layout_width="match_parent"
18         android:layout_height="wrap_content"
19         android:text="변환"
20         android:textSize="20dp"
21         android:textStyle="bold" />
22
23 </LinearLayout>
```

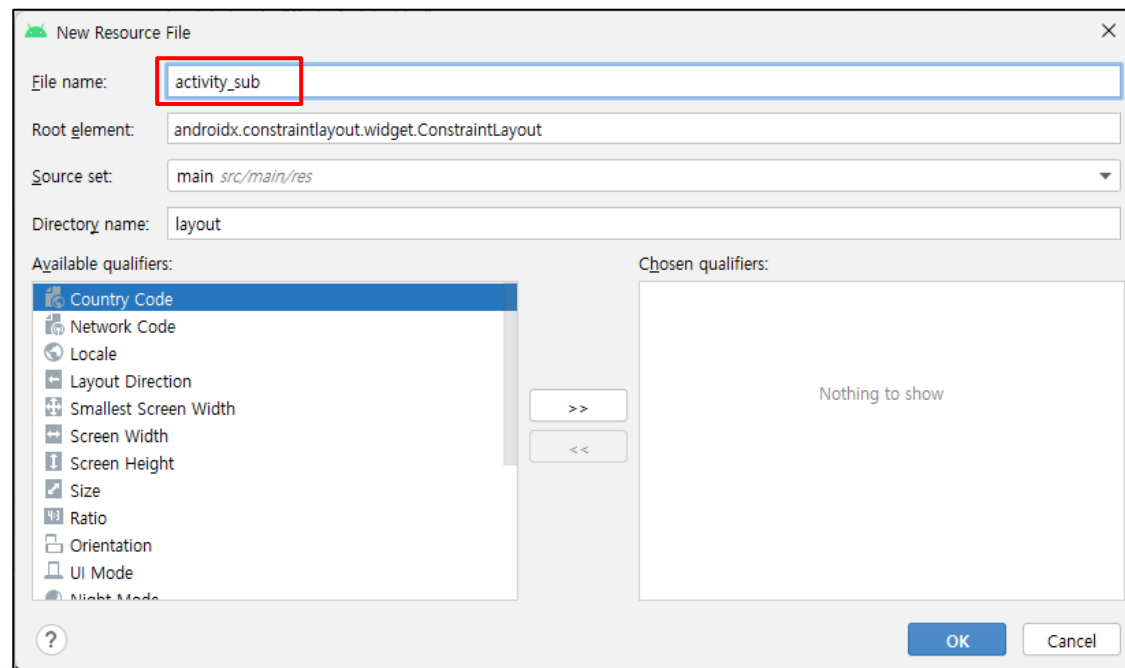
기초(따라하기) – 예제 1

2. activity_sub.xml 생성

- App -> res -> layout 파일 우 클릭 후 New
-> Layout Resource File

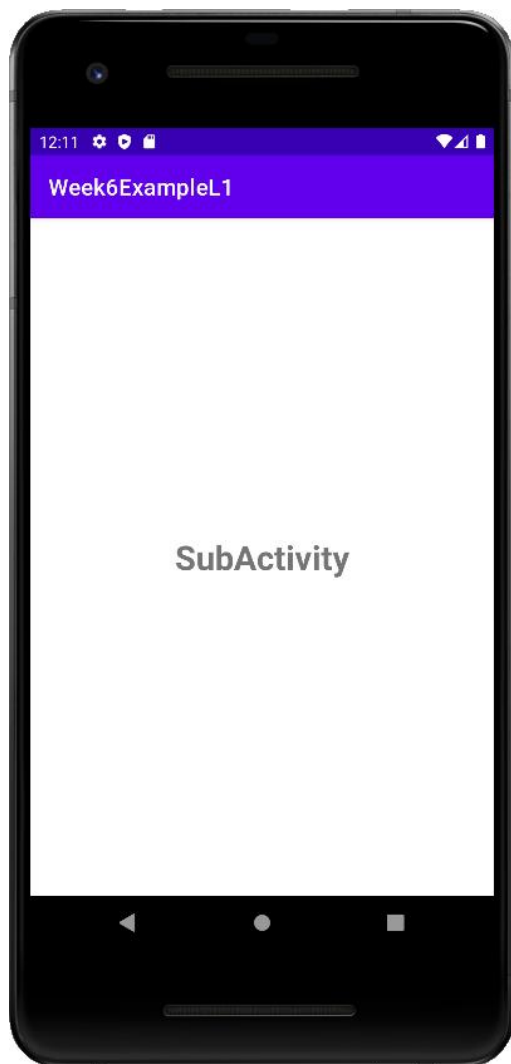


- File name 입력 후 엔터
- “activity_sub” 입력



기초(따라하기) – 예제 1

3. 서브 Activity 레이아웃 구성



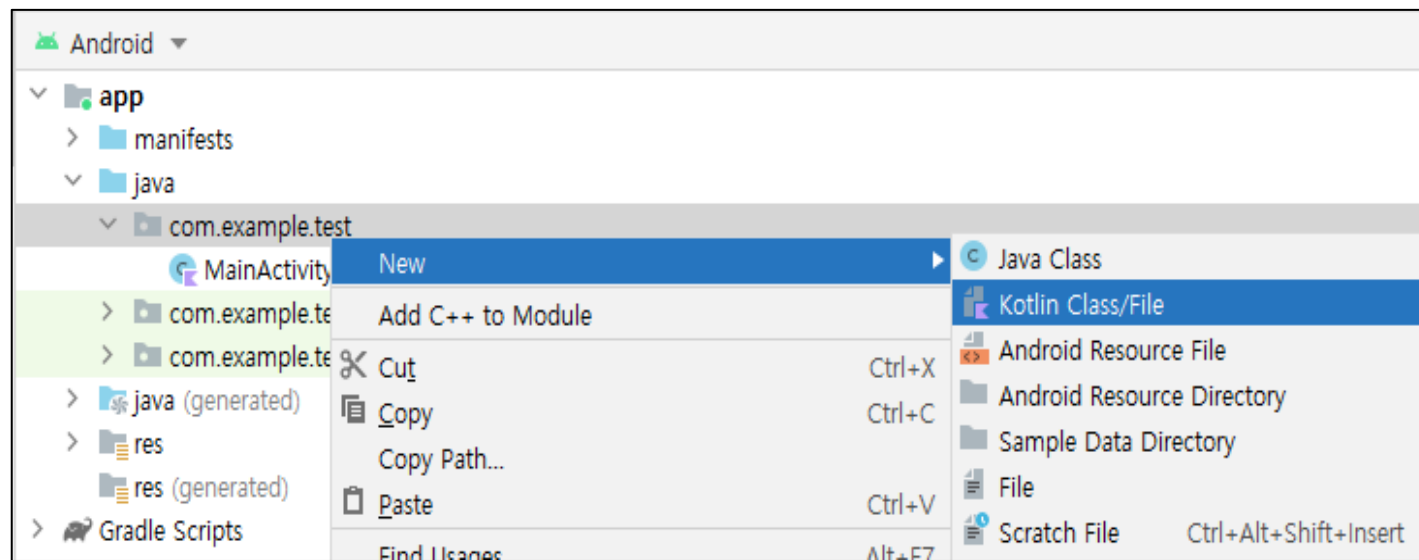
activity_sub.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6      <TextView
7          android:layout_width="match_parent"
8          android:layout_height="wrap_content"
9          android:layout_weight="5"
10         android:gravity="center"
11         android:text="SubActivity"
12         android:textSize="30dp"
13         android:textStyle="bold" />
14
15  </LinearLayout>
```

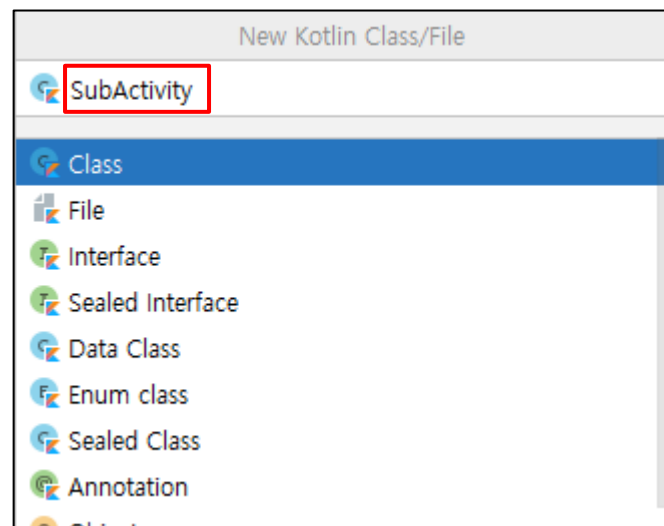

기초(따라하기) – 예제 1

4. SubActivity.kt 생성

- MainActivity가 있는 디렉토리에서
우클릭 -> New -> Kotlin/File



- Name 입력 후 엔터
- “SubActivity” 입력



기초(따라하기) – 예제 1

5. 서브 액티비티 구현

- (4 line) **AppCompatActivity()** : 액티비티의 서브 클래스인 AppCompatActivity로 생성한다.
- (6 line) **onCreate()**: 액티비티를 생성한다.
- (7 lines) **setContentView** : 뷰 데이터를 로드한다.

SubActivity.kt

```
1 package com.example.week6example12
2 import android.os.Bundle
3 import androidx.appcompat.app.AppCompatActivity
4 class SubActivity : AppCompatActivity() {
5     override fun onCreate(savedInstanceState: Bundle?) {
6         super.onCreate(savedInstanceState)
7         setContentView(R.layout.activity_sub)
8     }
9 }
```

기초(따라하기) – 예제 1

MainActivity.kt

6. Intent, 메인 액티비티 구현

- (11 line) **setContentView()**: 레이아웃 xml의 내용을 파싱하여 뷰를 생성한 후 속성을 설정 한다.
- (12 lines) **findViewById** : 레이아웃 xml에 있는 뷰 들의 id를 통해 뷰 객체를 생성한다.
- (14 line) **setOnClickListener**: 버튼 클릭 이벤트를 처리하는 리스너
- (15 line) **Intent**: intent 객체를 생성한다.
- (16 line) **startActivity()**: 새 액티비티를 실행한다.

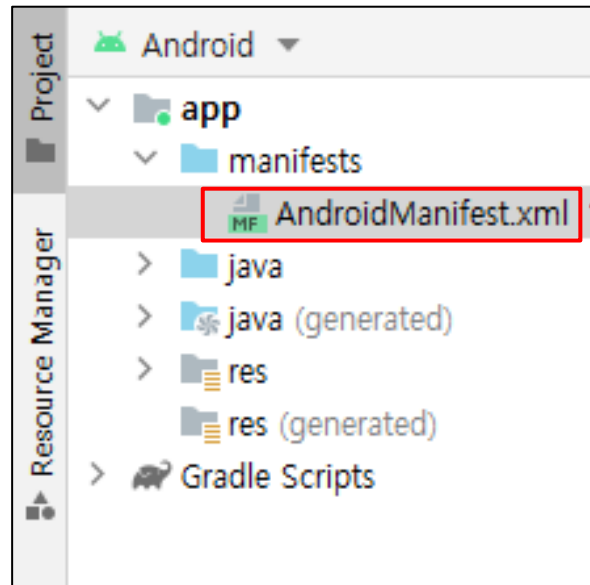
```
1 package com.example.week6example12
2 import android.app.Activity
3 import android.content.Intent
4 import androidx.appcompat.app.AppCompatActivity
5 import android.os.Bundle
6 import android.widget.Button
7
8 class MainActivity : AppCompatActivity() {
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12         val btn: Button = findViewById(R.id.btn_main)
13
14         btn.setOnClickListener { it: View!
15             val intent = Intent(packageContext: this, SubActivity::class.java)
16             startActivity(intent)
17         }
18     }
19 }
```

기초(따라하기) – 예제 1

7. 새로운 액티비티 등록(AndroidManifest.xml 수정)

AndroidManifest.xml

- (21 line) 서브 액티비티 등록하기



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3          package="com.example.week6">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="week6"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/Theme.Week6">
12
13         <activity
14             android:name=".MainActivity"
15             android:exported="true">
16             <intent-filter>
17                 <action android:name="android.intent.action.MAIN" />
18
19                 <category android:name="android.intent.category.LAUNCHER" />
20             </intent-filter>
21             <activity android:name=".SubActivity"></activity>
22         </activity>
23     </application>
24 </manifest>
```

작성

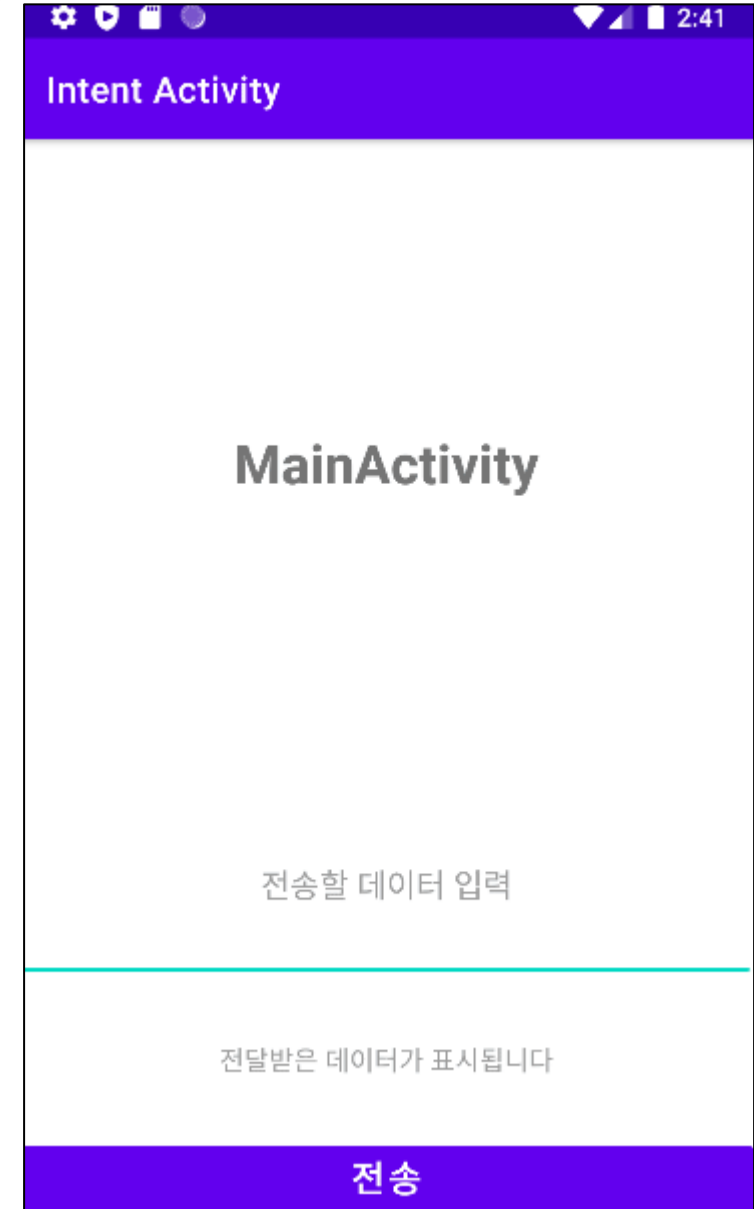
- 참고사항

- AndroidManifest.xml : 프로젝트 요소 및 권한 등을 관리하는 파일

응용(로직구현) – 예제 2

❖ 양방향 Activity

1. activity_main.xml 작성
2. activity_sub.xml 작성
3. MainActivity.kt 작성
4. SubActivity.kt 작성



1. activity_main.xml 작성

- 전송할 데이터 입력하는 EditText 추가
- 전송 받은 데이터 표시할 TextView 추가
- 데이터 전송 시 누를 버튼 추가
- 뷰 사이 비율을 위한 웨이트 추가

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6      <TextView
7          android:layout_width="match_parent"
8          android:layout_height="wrap_content"
9          android:layout_weight="5"
10         android:gravity="center"
11         android:text="MainActivity"
12         android:textSize="30dp"
13         android:textStyle="bold" />
14
15     <EditText
16         android:id="@+id/ev_main"
17         android:layout_width="match_parent"
18         android:layout_height="wrap_content"
19         android:layout_weight="1"
20         android:gravity="center"
21         android:hint="전송할 데이터 입력"
22         android:singleLine="true" />
23
24     <TextView
25         android:id="@+id/res_main"
26         android:layout_width="match_parent"
27         android:layout_height="wrap_content"
28         android:layout_weight="1"
29         android:gravity="center"
30         android:hint="전달받은 데이터가 표시됩니다"
31         android:textSize="15dp" />
32
33     <Button
34         android:id="@+id/btn_main"
35         android:layout_width="match_parent"
36         android:layout_height="wrap_content"
37         android:text="전송"
38         android:textSize="20dp"
39         android:textStyle="bold" />
40
41 </LinearLayout>
```

2. activity_sub.xml 작성

- 전송할 데이터 입력하는 EditText 추가
- 전송 받은 데이터 표시할 TextView 추가
- 데이터 전송 시 누를 버튼 추가
- 뷰 사이 비율을 위한 웨이트 추가

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical">
6      <TextView
7          android:layout_width="match_parent"
8          android:layout_height="wrap_content"
9          android:layout_weight="5"
10         android:gravity="center"
11         android:text="SubActivity"
12         android:textSize="30dp"
13         android:textStyle="bold" />
14
15     <EditText
16         android:id="@+id/ev_sub"
17         android:layout_width="match_parent"
18         android:layout_height="wrap_content"
19         android:layout_weight="1"
20         android:gravity="center"
21         android:hint="전송할 데이터 입력"
22         android:singleLine="true" />
23
24     <TextView
25         android:id="@+id/res_sub"
26         android:layout_width="match_parent"
27         android:layout_height="wrap_content"
28         android:layout_weight="1"
29         android:gravity="center"
30         android:text="전달받은 데이터가 표시됩니다"
31         android:textSize="15dp" />
32
33     <Button
34         android:id="@+id/btn_sub"
35         android:layout_width="match_parent"
36         android:layout_height="wrap_content"
37         android:text="전송"
38         android:textSize="20dp"
39         android:textStyle="bold" />
40
41 </LinearLayout>
```

응용(로직구현) – 예제 2

MainActivity.kt

3. MainActivity.kt 작성

- 전송 버튼 클릭 시 EditText가 비어 있을 경우, 아무 동작 없이 토스트 메시지만 출력할 수 있도록 구현
- 전송할 데이터를 Intent에 포함할 putExtra 함수 추가
 - putExtra(name, value): Intent를 통해 전달할 데이터를 데이터(value)에 대한 라벨(name)과 함께 Intent에 추가한다.
- 메시지를 보낸 후 다시 메인 액티비티로 돌아왔을 때, EditText는 비워져 있도록 구현

```
14 class MainActivity : AppCompatActivity() {
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         setContentView(R.layout.activity_main)
18         val btn: Button = findViewById(R.id.btn_main)
19         val ev = findViewById<EditText>(R.id.ev_main)
20         val res: TextView = findViewById(R.id.res_main)
21
22         btn.setOnClickListener { it: View!
23             if (ev.text.isEmpty()) {
24                 Toast.makeText(context: this, text: "값을 입력 해주세요.", Toast.LENGTH_SHORT).show()
25                 return@setOnClickListener
26             }
27             val data: String = ev.text.toString()
28             val intent = 
29             intent.putExtra(
30             startActivityForResult(intent, requestCode: 100)
31             ev.text = null
32         }
33     }
34
35     override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
36         super.onActivityResult(requestCode, resultCode, data)
37         val res: TextView = findViewById(R.id.res_main)
38         if (resultCode == Activity.RESULT_OK) {
39             when (requestCode) {
40                 100 -> {
41                     res.text = data!!.getStringExtra(name: "data").toString() + "\n-send from sub"
42                 }
43             }
44         }
45     }
46 }
47 }
```


응용(로직구현) – 예제 2

3. MainActivity.kt 작성

- SubActivity에서 응답을 받을 onAcitivityResult 구현
 - onActivityResult()의 파라미터는 requestCode, resultCode, data(Intent) 세가지로 구성
 - requestCode는 startActivityForResult()함수의 파라미터에 포함되어 다음 액티비티로 전송된 후, 해당 액티비티가 종료되면 Intent에 자동으로 포함되어 반환
이를 통해 어떤 액티비티로 부터 온 Intent인지 구분함
 - resultCode는 액티비티의 결과가 성공했는지 실패했는지 구분함(RESULT_OK, RESULT_CANCEL), 이전 액티비티에서 setResult()를 통해 값을 전달 받음
 - getStringExtra(name): finish() 된 액티비티 에서의 값(String)을 받아오며, 데이터의 라벨(name)에 해당하는 데이터를 받을 수 있음

응용(로직구현) – 예제 2

4. SubActivity.kt 작성

- MainActivity.kt의 코드를 참조하여 구현할 것!
 - Finish() : 현재 Activity 종료
- 새로운 액티비티 등록

AndroidManifest.xml

```
21 <activity android:name=".SubActivity"></activity>
```

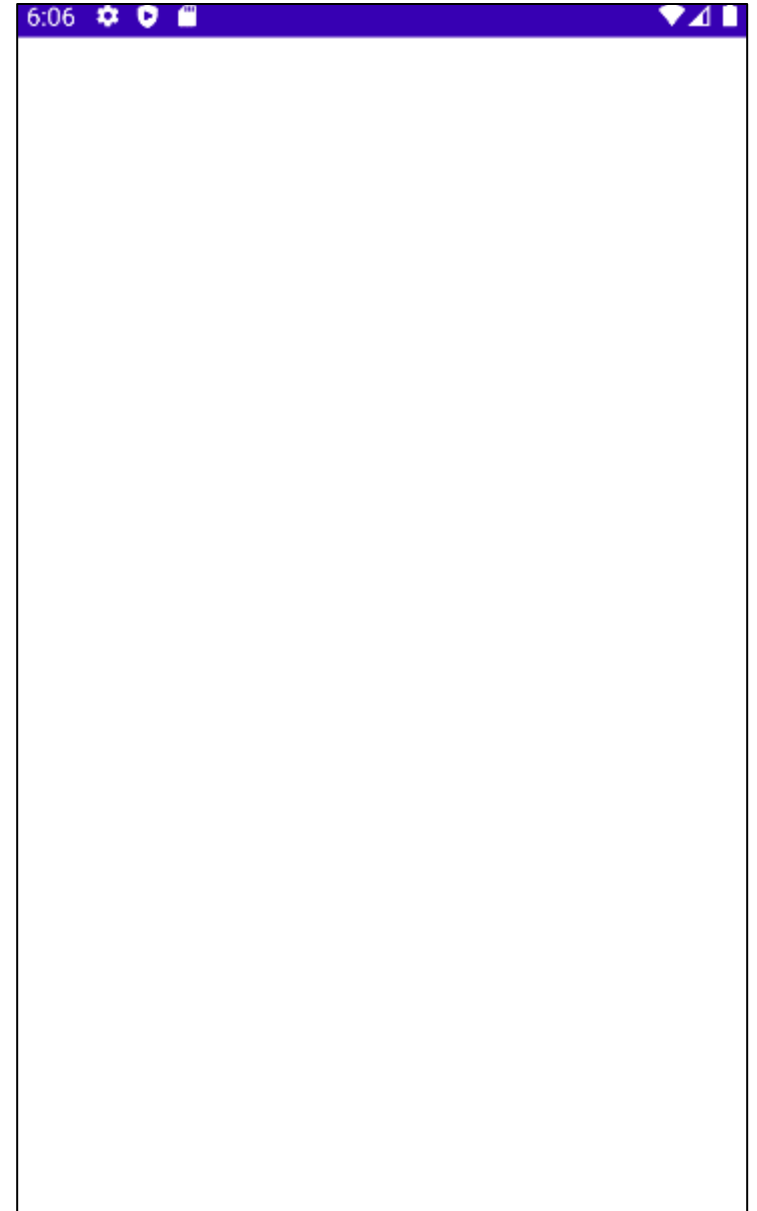
SubActivity.kt

```
class SubActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
  
    }  
  
    btn.setOnClickListener { it: View!  
  
    }  
    finish()  
}
```

심화(완성하기) – 예제 3

❖ 연락처 만들기

1. 연락처 만들기 (1단계)
2. 연락처 만들기 (2단계)



심화(완성하기) – 예제 3

❖ 연락처 만들기 (1단계)

■ 조건

- 이름, 나이, 전화번호, 주소, 기타 항목들을 입력할 수 있도록 구현
 - TextView는 hint속성을 통해 무슨 칸인지 알 수 있도록 한다.
- 저장 버튼을 눌러 입력한 정보들을 SubActivity에서 확인할 수 있도록 구현
- SubActivity에서 되돌아가기 버튼을 통해 다시 MainActivity로 돌아온다.
- 단, 돌아왔을 경우 값이 초기화 되어야함
 - TextView는 hint속성을 통해 무슨 칸인지 알 수 있도록 한다.

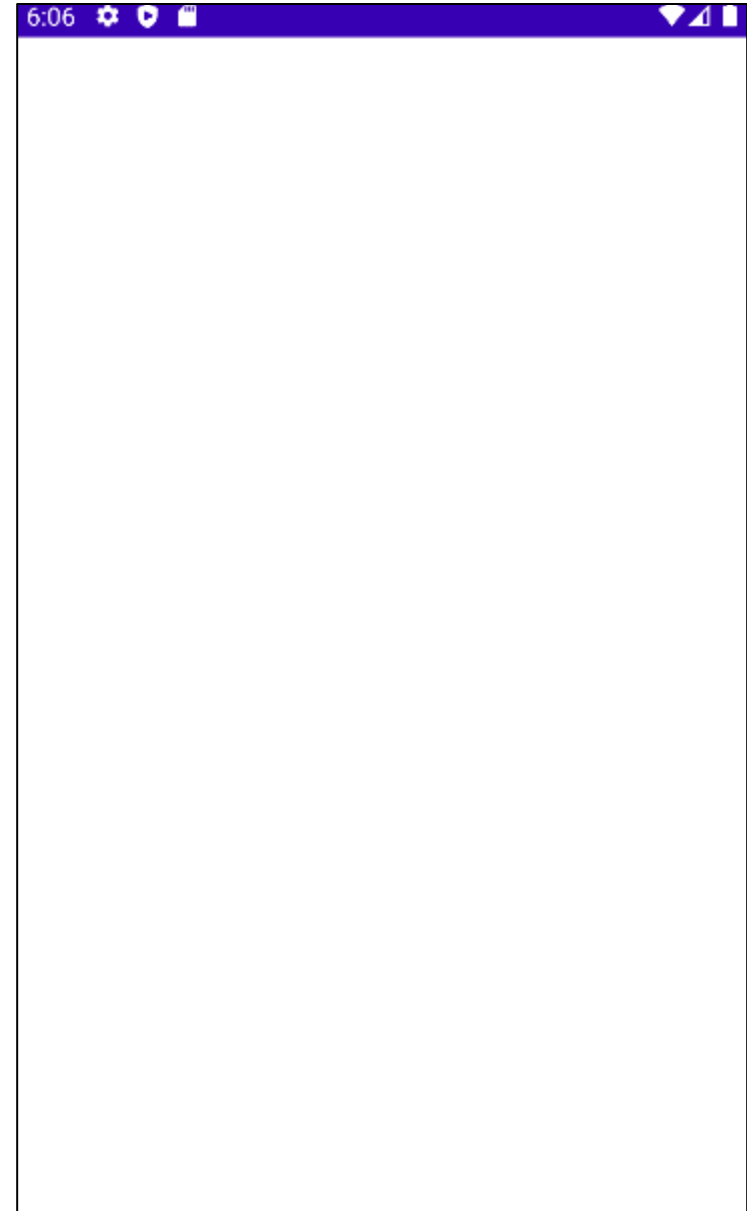


심화(완성하기) – 예제 3

❖ 연락처 만들기 (2단계)

■ 조건

- ImageView를 통해 사진을 넣을 수 있게 하며, 이미지 뷰 클릭 시 갤러리와 연동될 수 있도록 인텐트로 구현
 - Image 뷰에 대한 객체 생성 후 .isClickable값을 true 로 지정
 - 갤러리의 Intent 생성 시 ACTION_GET_CONTENT 사용
 - setType("image/*"), requestCode: GALLERY = 1
- ImageView의 사진을 **Bitmap**으로 받아와서 300x300 크기로 변환 및 **Bitmap**을 사용하여 다음 인텐트로 전달
 - Bitmap.createScaledBitmap() 함수 사용
 - getParcelableExtra -> Retrieve extended data from the intent
- SubActivity에 완료 버튼과 수정버튼을 추가하여 완료 버튼을 누르면 결과 화면으로 갈 수 있도록 제작
- 이때 두 버튼 모두 **MainActivity**로 돌아가야 하며, 수정을 눌렀을 경우엔 사전에 작성한 **내용이 모두 포함되어야** 하며 **완료**를 눌렀을 경우 작성된 **내용은 모두 지워져야** 함
 - 이미지 초기화 setImageResource() 함수 사용
- ResultActivity.kt, activity_result.xml 를 추가하여 최종 저장되는 모습을 보여주도록 한다. 이때, 정상적으로 저장되었다는 메시지와 함께 짧은 토스트 메시지를 출력



심화(완성하기) – 예제 3

❖ 연락처 만들기 (2단계)

■ 이미지 관련 참고 코드

- `isClickable`: 이미지 뷰를 클릭할 수 있게 함
- `setType()`: 인텐트의 형식을 지정해줌
 - “image/*”: URI 반환
- `GALLERY`: 갤러리에 대한 Intent 임을 구별해줌
- `resizeBitmap()`: 비트맵의 높이와 넓이를 `.createScaledBitmap`을 통해 재조정 하는 함수를 구현할 것
 - 해당 함수는 저장버튼을 통해 저장한 후 SubActivity에서 본인이 선택한 이미지를 보여주기 위해 구현한다.
 - `resize`크기는 `width = 300, height = 300`이며, 해당 과정을 진행하지 않을 경우 인텐트를 통해 이미지를 SubActivity로 전달할 때 용량초과로 에러가 발생할 수도 있다.

MainActivity.kt

```
val photo: ImageView =    
photo.isClickable = true  
photo.  { it: View!  
    openGallery()  
}
```

```
private fun openGallery() {  
    val intent: Intent = Intent(Intent.ACTION_GET_CONTENT)  
    intent.setType( )  
    startActivityForResult(intent, GALLERY)  
}
```

```
private fun resizeBitmap(bitmap: Bitmap, width: Int, height: Int): Bitmap {  
    return Bitmap.createScaledBitmap(  
        bitmap,  
        높이,  
        넓이  
        filter: false  
    )  
}
```

```
val resize = resizeBitmap(bitmap, width: 300, height: 300)  
intent.putExtra( name: "image", resize)
```

심화(완성하기) – 예제 3

❖ 연락처 만들기 (2단계)

■ 이미지 관련 참고 코드

- MediaStore.*media-type*.Media: 외부 저장소볼륨을 검색하고 *media-type*에 따라 추가할 미디어의 타입을 정의함
 - *media-type*: 사진과 스크린샷을 포함
 - DCIM/ 및 Picture/ 디렉터리에 저장
- setImageBitmap(bitmap): 해당 뷰에 비트맵 이미지를 설정함
- Intent.getParcelableExtra: Intent로 부터 확장된 데이터를 가져옴

» Ref:
[https://developer.android.com/reference/android/content/Intent#getParcelableExtra\(java.lang.String\)](https://developer.android.com/reference/android/content/Intent#getParcelableExtra(java.lang.String))

MainActivity.kt

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == REQUEST_IMAGE_CAPTURE) {
        when (resultCode) {
            RESULT_OK -> {
                var imageData: Uri? = data?.data
                try {
                    val bitmap = MediaStore.Images.Media.getBitmap(contentResolver, imageData)
                    val pt: ImageView = findViewById(R.id.photo)
                    pt.setImageBitmap(bitmap)
                } catch (e: Exception) {
                    e.printStackTrace()
                }
            }
        }
    }
}
```

MainActivity.kt

```
val bitmap = data!!.getParcelableExtra<Bitmap>(name: "image")
```

SubActivity.kt

```
val bitmap = intent.getParcelableExtra<Bitmap>(name: "image")
```

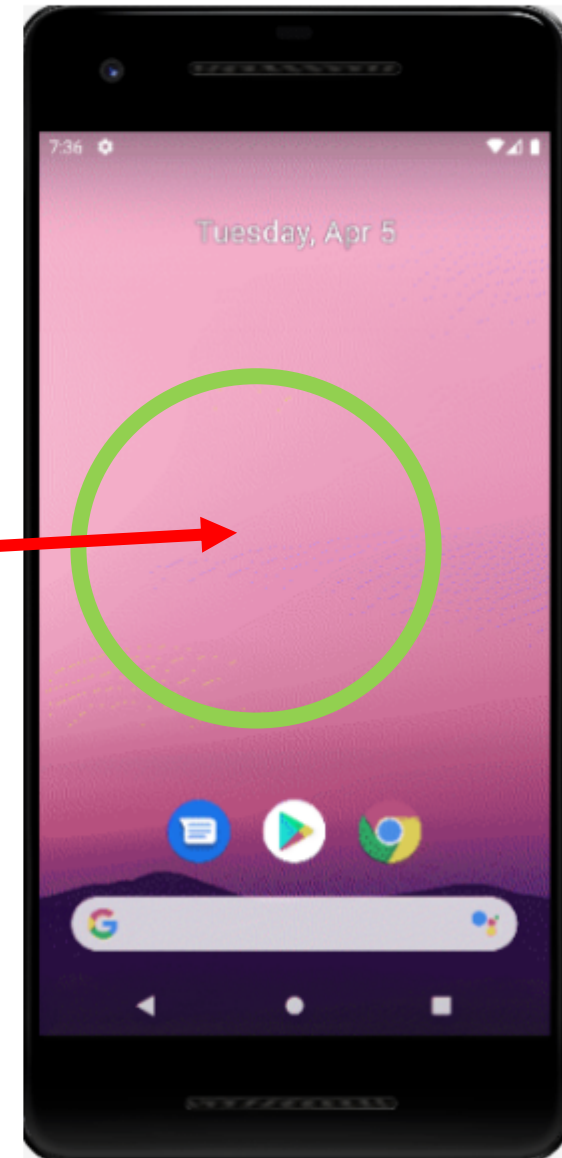
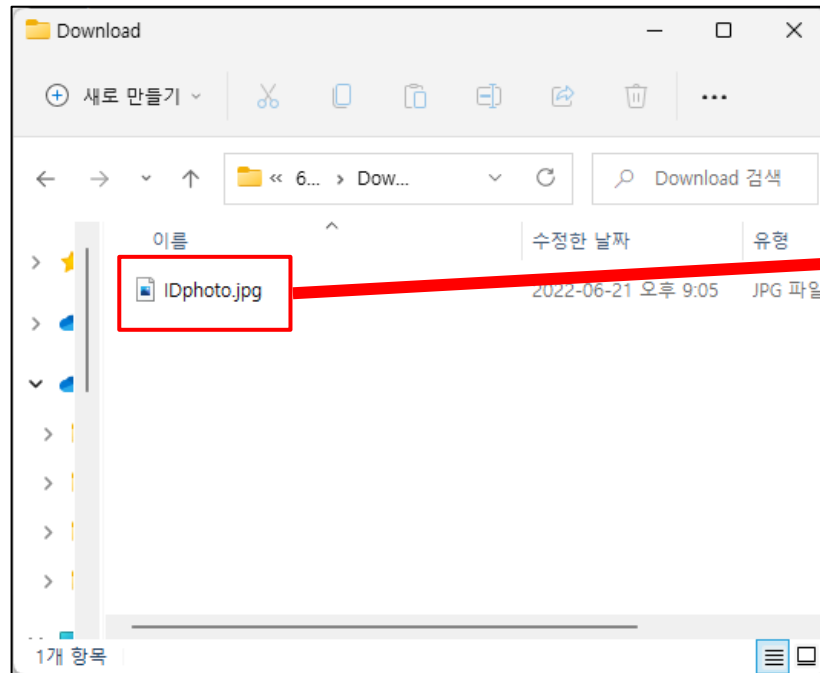
ResultActivity.kt

```
val photo = intent.getParcelableExtra<Bitmap>(name: "image")
```

심화(완성하기) – 예제 3

❖ 연락처 만들기 (2단계)

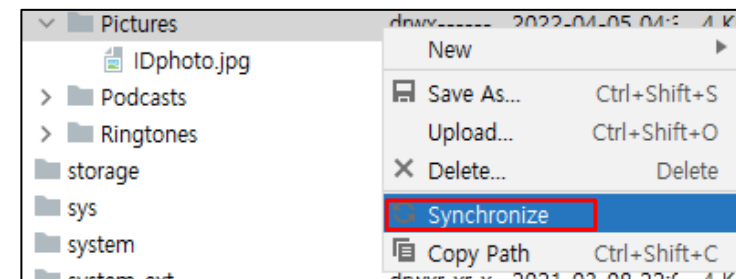
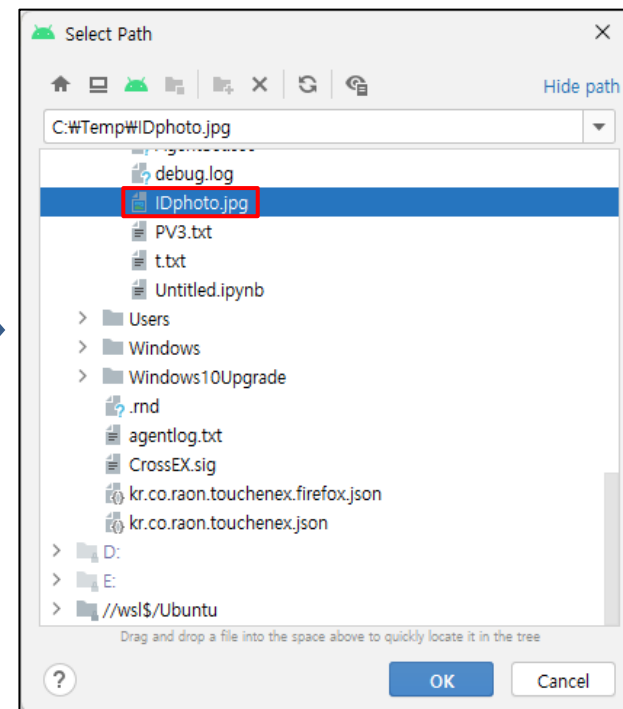
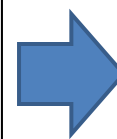
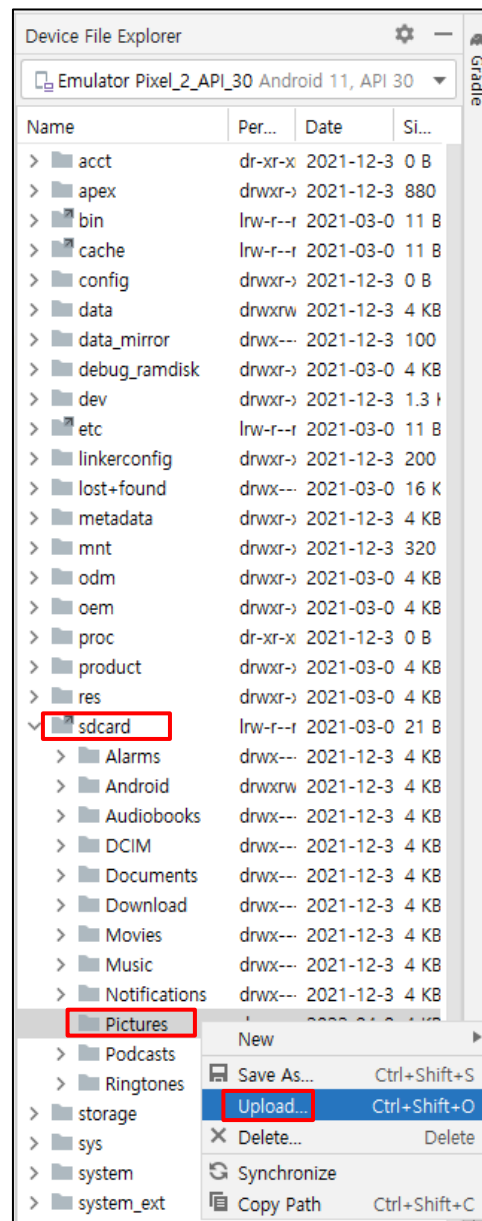
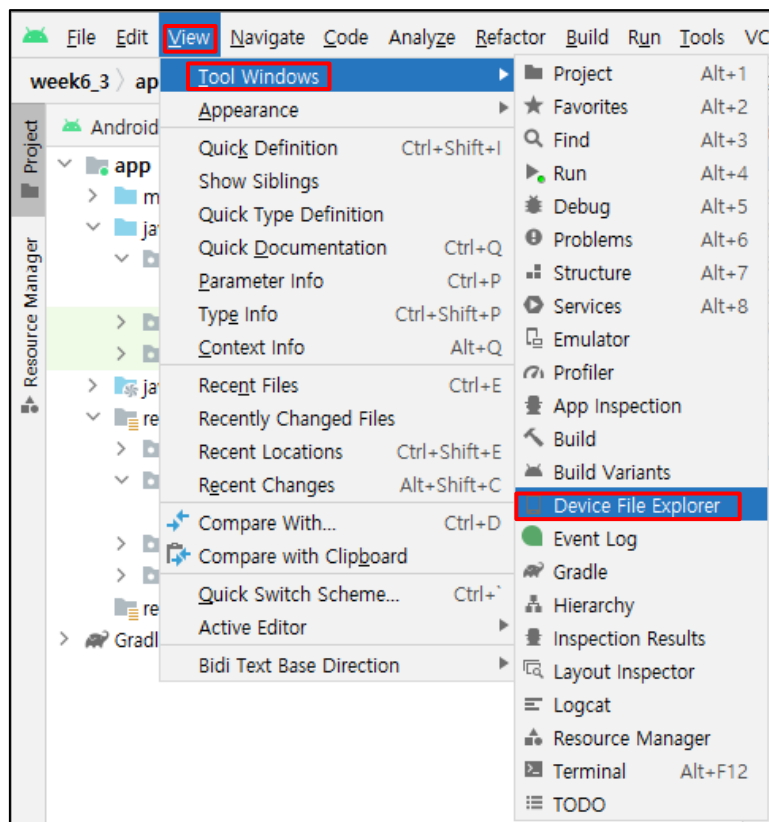
- AVM 에 사진 넣는 방법 1
 - 에뮬레이터의 홈 화면의 빈 공간으로 해당 파일을 드래그하기
 - Download 폴더에 해당 파일이 생성됨



심화(완성하기) – 예제 3

❖ 연락처 만들기 (2단계)

- AVM 에 사진 넣는 방법 2
 - 업로드 하기
 - 동기화가 잘 안되나, 원하는 위치 지정 가능

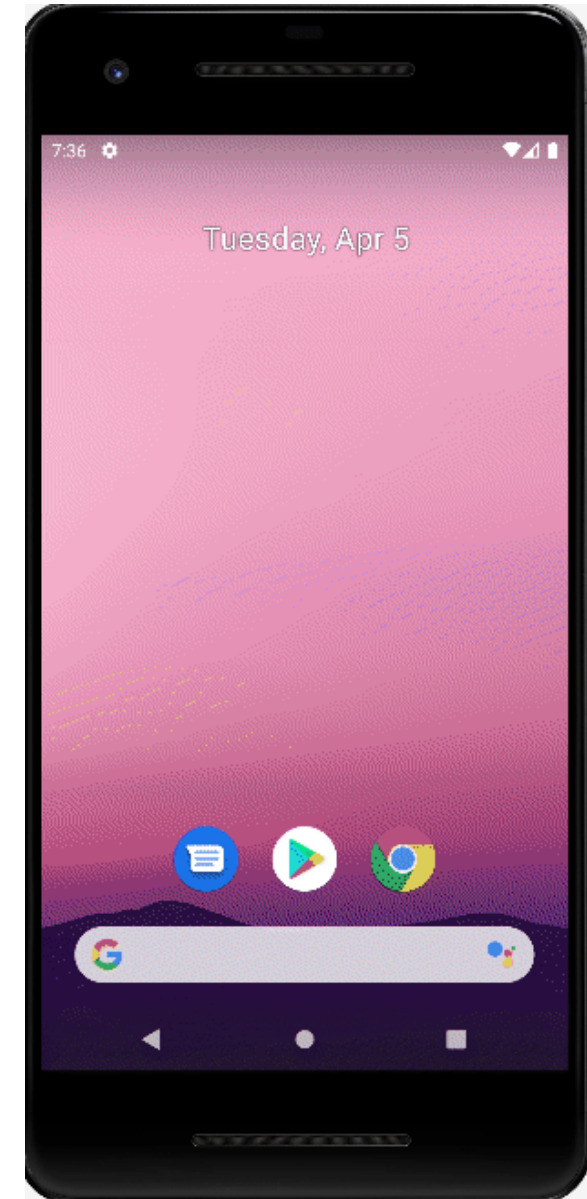


심화(과제물) – 예제 4

❖ 자기소개서 만들기

■ 결과물

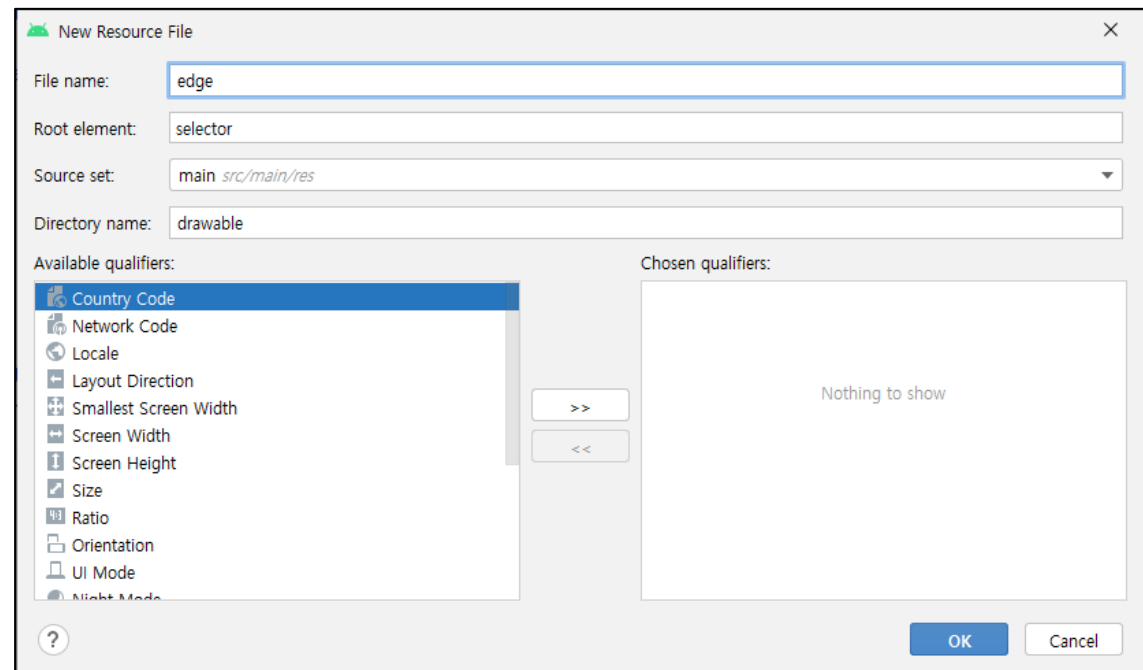
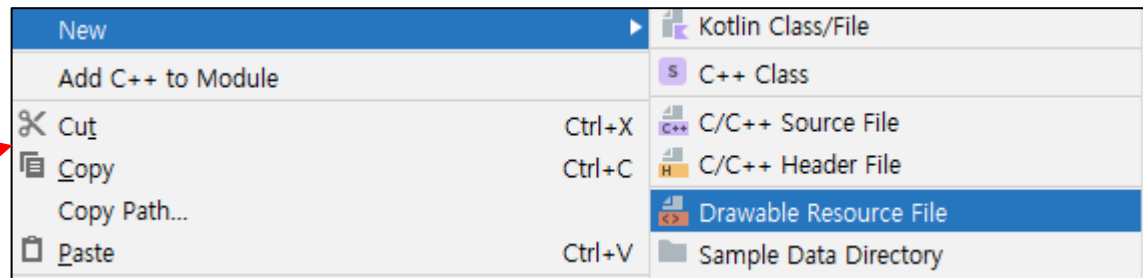
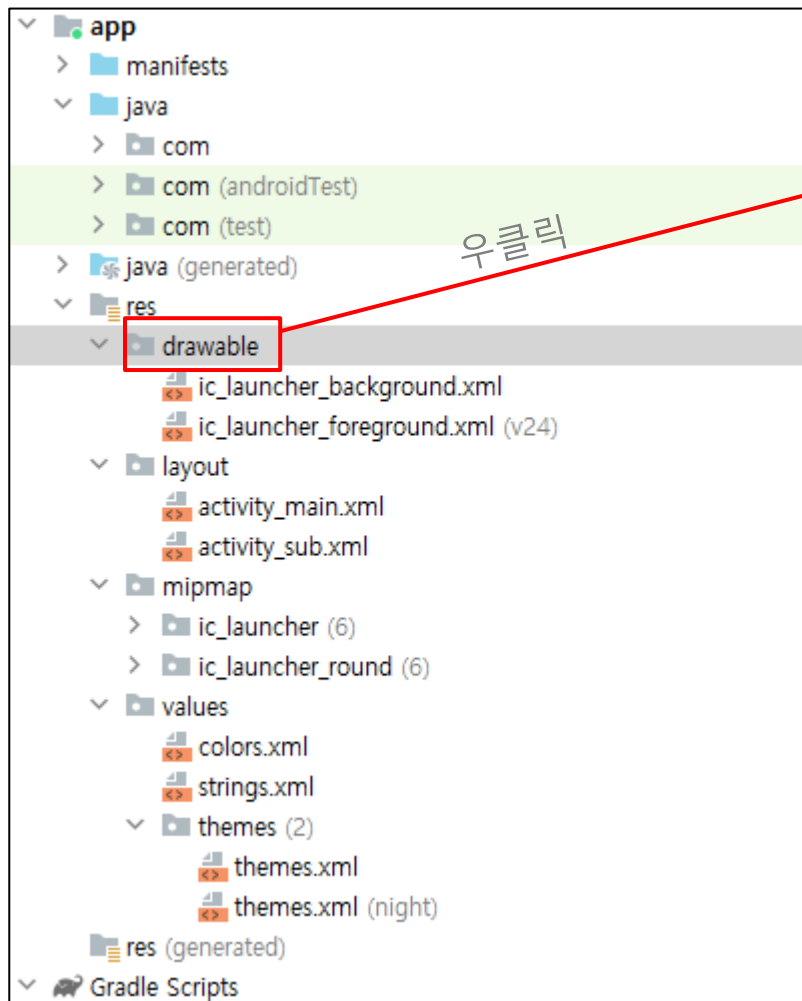
- Main Activity: 이력서
 - 본인 사진, 이름, 생년 월일, 연락처, 이메일, 주소, 최종 학력, 전공, 특이사항 등
 - 본인 사진 입력 시 예제 3-2와 동일한 방법 사용할 것
 - ScrollView를 이용하여 예시와 같이 구현할 것
 - 버튼을 통해 SubActivity로 전환
- Sub Activity : 이력서
 - ScrollView 를 사용하여 구현할 것
 - 성장과정, 학창시절, 성격 장단점, 핵심역량
 - 버튼은 스크롤을 내리더라도 하단에 고정되어야 함
- 버튼을 통한 액티비티 변경
 - 액티비티 하단에, 버튼을 통해 두 액티비티 전환 => 양방향



심화(과제물) – 예제 4

❖ 자기소개서 만들기

- EditText 테두리 추가하는 방법 [1/2]



심화(과제물) – 예제 4

❖ 자기소개서 만들기

■ EditText 테두리 추가하는 방법 [2/2]

- (2 Line) <layer-list>: <item>요소를 중첩해서 사용
- (4 Line) <item>: Drawable 속성값을 통해 Drawable 리소스를 정의하는데 사용
- (5 Line) <shape>: <item>의 모양을 정의하는 태그
- (6 Line) <solid>: <item>의 색상을 선택
- (11~14 Line) item의 크기 조절

```
Edge.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <layer-list xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <item>
5          <shape android:shape="rectangle">
6              <solid android:color="#000000" />
7          </shape>
8      </item>
9
10     <item
11         android:bottom="1dp"
12         android:left="1dp"
13         android:right="1dp"
14         android:top="1dp">
15
16         <shape android:shape="rectangle">
17             <solid android:color="#ffffff" />
18         </shape>
19
20     </item>
21 </layer-list>
```

Activity_main.xml

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_row="2"
    android:layout_column="0"
    android:layout_weight="1"
    android:background="@drawable/edge"
    android:gravity="center" />
```



| 기간 | 학교 | 학과 | 비고 |
|----|----|----|----|
| | | | |
| | | | |
| | | | |
| | | | |

<실행 결과>