Fragment by androidx

Mobile Software
2019 Fall

Native Fragment vs. Support Fragment vs. Androidx Fragment (2019. 1. 22 포스팅 발췌)

- Fragment 배우고 싶은데 library가 이렇게 많은 거야? 뭘 선택해야 해???

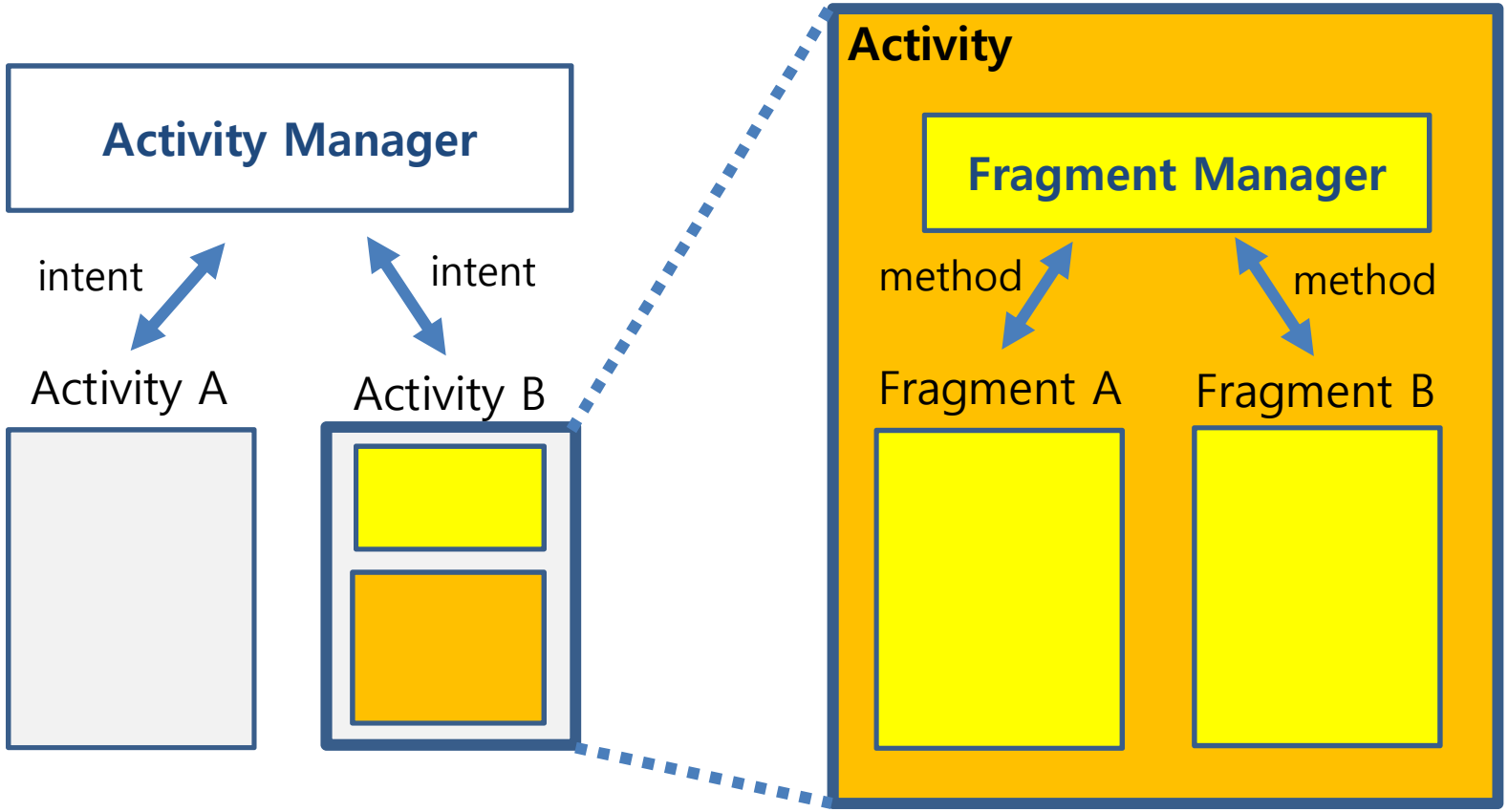
Library	Package
Support Library	android.support.v4.app.Fragment
AndroidX Library	androidx.fragment.app.Fragment
Native	android.app.Fragment

  - **Support library**
    - Fragment 는 Android 3 (API 11)부터 도입. 그럼 그 이전 버전을 사용하는 device는?
    - Google은 이전 버전을 사용 중인 device를 지원하기 위해 support library를 제공
  - **Native library**
    - Android 3+ device는 fragment를 지원하는 기능을 당연히 갖고 있겠지
    - 그런데, fragment는 버전이 바뀔 수록 좋아져… Android 3 < Android 5 < …
      - 안드로이드는 API 28부터 native library 지원을 중단하기로 선언해.
    - 따라서, support library를 사용하는 코드를 그동안 많이 봐왔겠지…
  - **Android X library**
    - Google은 Jetpack 컴포넌트를 추가하면서
    - support library보다 성능이 뛰어난 라이브러리를 새로 만들었어.
    - 그게 바로 android x 야… 따라서 Jetpack도 꼭 알아야만 하겠지!

https://www.andreasschrade.com/android-native-fragment-or-support-fragment-or-androidx-fragment

Fragment (1/2)

- Fragment는 Activity의 하나의 조각



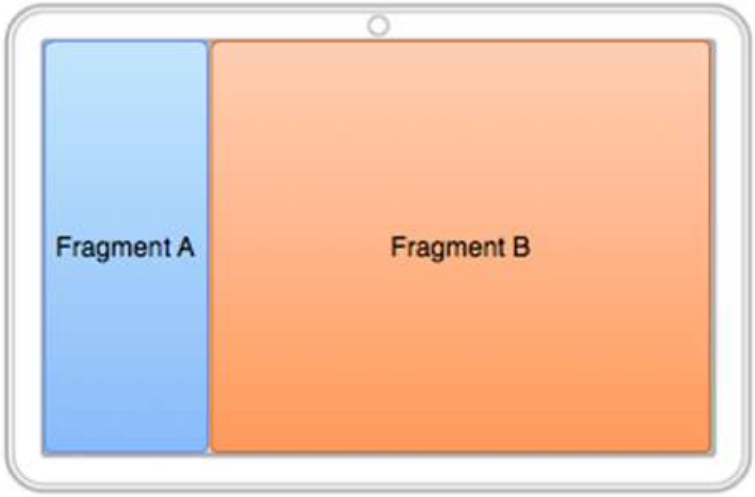
Fragment (2/2)

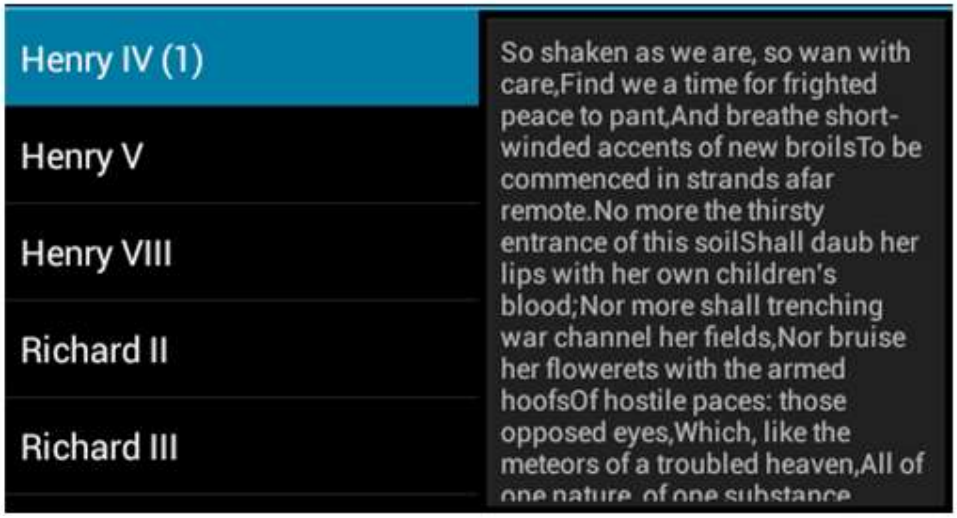
- A Fragment represents a behavior or a portion of user inter face in an Activity.

- You can **combine multiple fragments in a single activity**
  - to build a multi-pane UI and reuse a fragment in multiple acti vities.

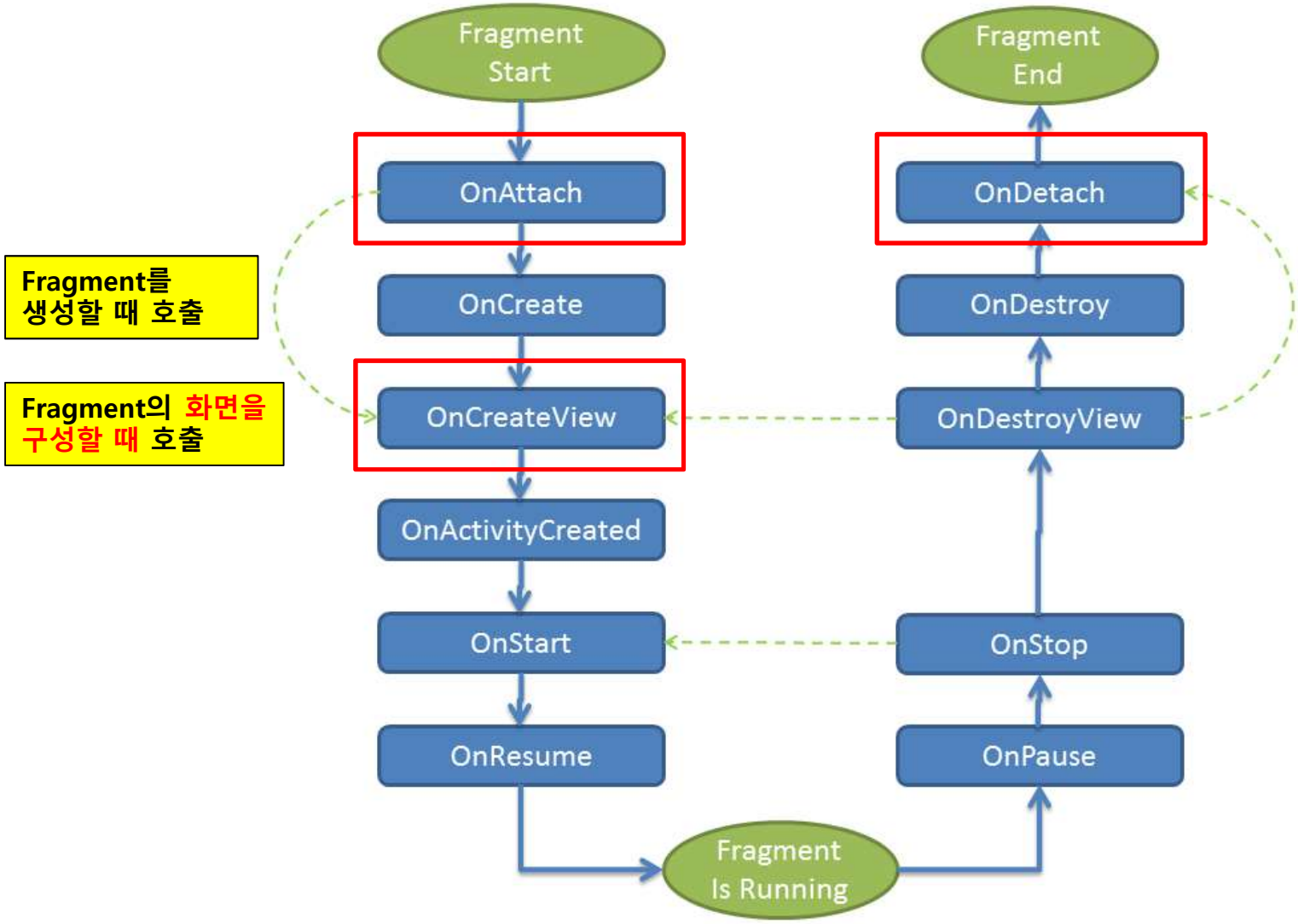
- You can think of a fragment as a modular section of an acti vity,
  - which **has its own lifecycle, receives its own input events, a nd which you can add or remove while the activity is runni ng**
  - sort of like a "**sub activity**" that you can reuse in different acti vities.

Fragment Idea





Fragment Lifecycle



실습 1: 간단한 Fragment

Fragment 구현

Fragment는 layout 파일과 Layout을 화면에 출력하기 위한 소스 파일 등 2개 파일로 구성

FragmentA.kt

```
class FragmentA : Fragment()
```

fragment_a.xml

activity_main.xml

```
<fragment
    android:id="@+id/fragment"
    android:name="edu.incheon.ch9project.FragmentA"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:layout="@layout/fragment_a" />
```

Activity의 layout에 view로 포함

실습 1: Fragment A - Layout

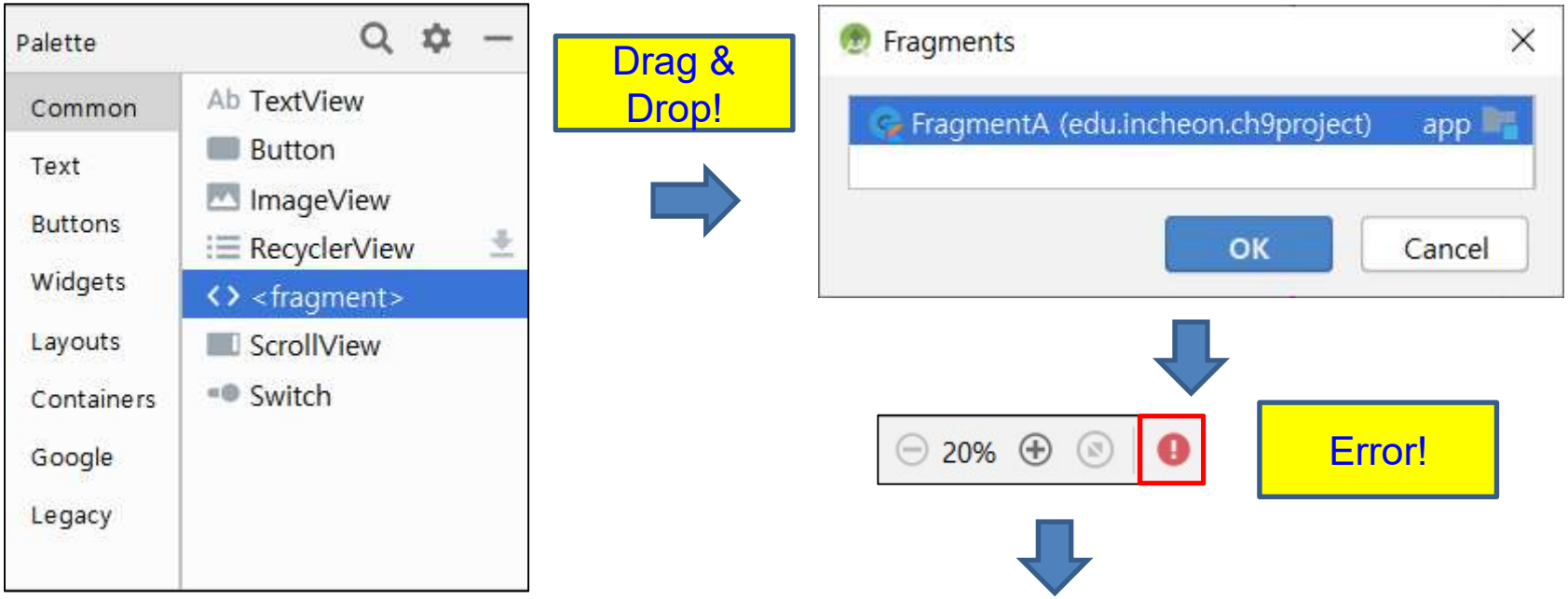
res/layout/fragment_a.xml

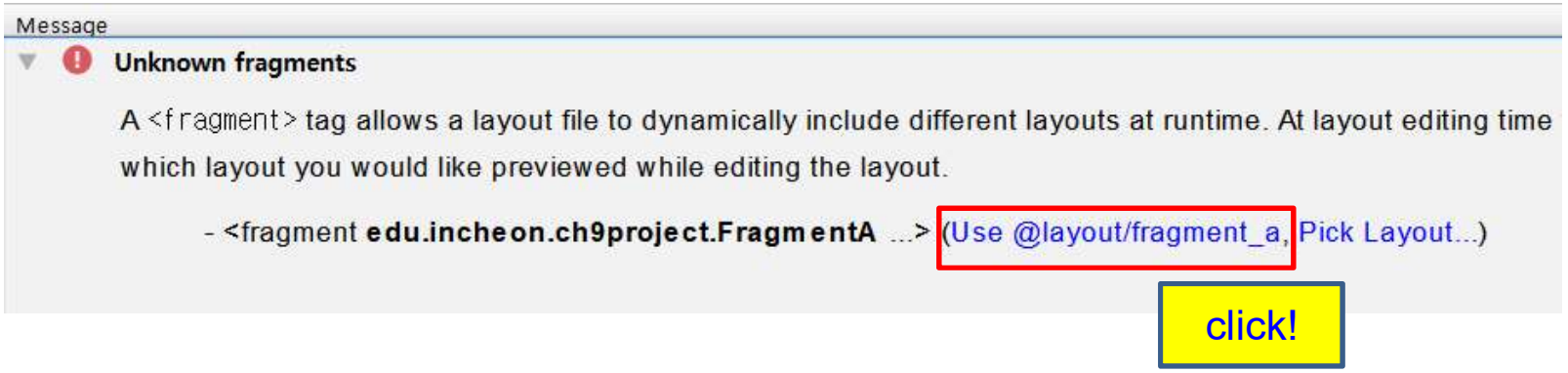
```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_light">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Fragment A" />
</RelativeLayout>
```

소스코드 – 1쪽

FragmentA 를 layout (activity_main) 에 추가





실습 1: Fragment A

FragmentA.kt

```
import androidx.fragment.app.Fragment

class FragmentA : Fragment() {
    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_a,
            container, false)
    }
}
```

소스코드 – 1쪽

잠깐! LayoutInfater.inflate( )

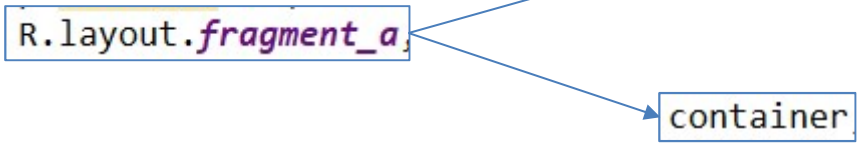
아래 2개 문장의 차이점은 무엇일까?

```
val view = inflater.inflate(R.layout.fragment_a,
    container, false)
```

```
val view = inflater.inflate(R.layout.fragment_a,
    container, true)
```

```
View inflate (XmlPullParser parser,
                ViewGroup root,
                boolean attachToRoot)
```

3번째 parameter가 true일 경우 생략 가능



3번째 parameter가 false일 경우
view가 fragment_a의 root view

3번째 parameter가 true일 경우
container가 fragment_a의 root view

실습 1: Activity 레이아웃에 fragment 추가

res/**layout**/**activity_main.xml**

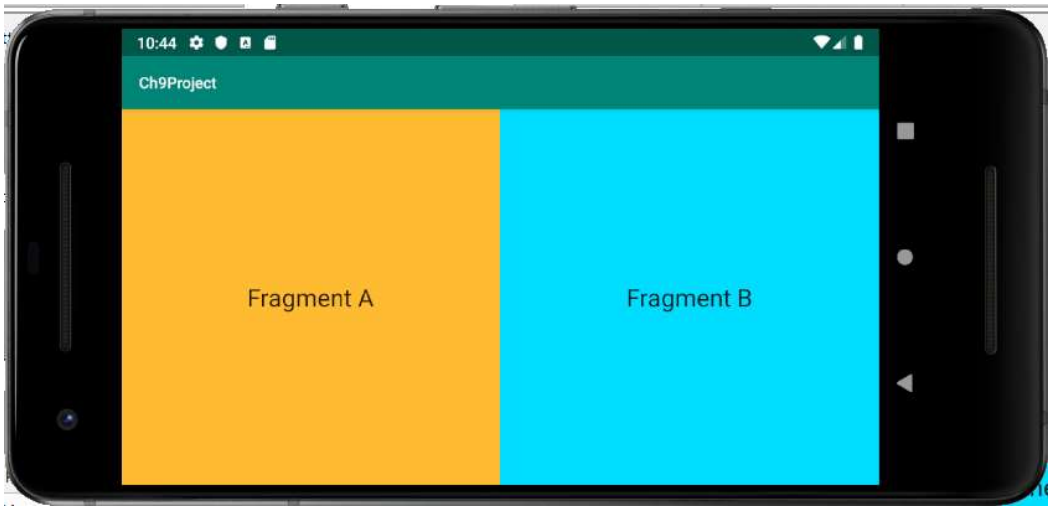
```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

    <fragment
        android:id="@+id/fragment"
        android:name="edu.ourincheon.ch9project.FragmentA"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout="@layout/fragment_a" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

android : name → fragment를 정의한 클래스 파일
tools : layout → fragment의 레이아웃 파일

소스코드 – 1~2쪽

실습 2: 하나의 Activity에 2개의 Fragment



Device 90도 회전
(landscape mode)

실습 2: setup

- Fragment 레이아웃
  - **fragment_a.xml, fragment_b.xml**
- Fragment 클래스 만들기
  - **FragmentA.kt, FragmentB.kt**
- **Fragment를 Activity에 추가**
  - **2개 Activity 레이아웃 작성**
    - **activity_main.xml, activity_main.xml(land)**
- Activity 작성
  - **MainActivity.kt**

실습 2: Activity 레이아웃에 fragments 추가

res/layout/activity_main.xml

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

    <fragment
        android:id="@+id/fragment"
        android:name="edu.ourincheon.ch9project.FragmentA"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_a" />
    <fragment
        android:id="@+id/fragment2"
        android:name="edu.ourincheon.ch9project.FragmentB"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_b" />
</LinearLayout>
```

소스코드 – 3~4쪽

실습 2: Activity 레이아웃에 fragments 추가

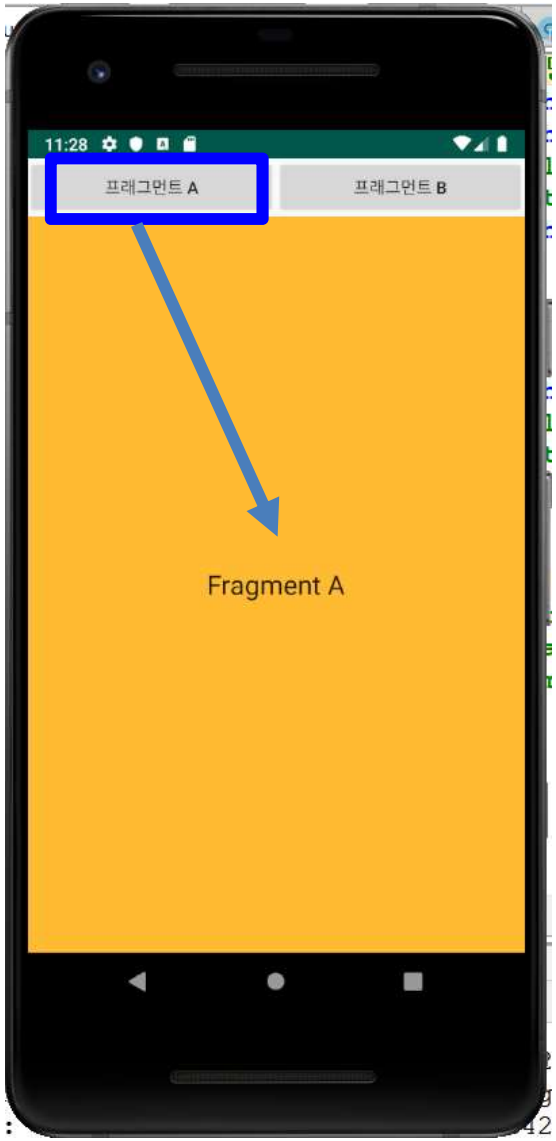
res/layout-land/activity_main.xml

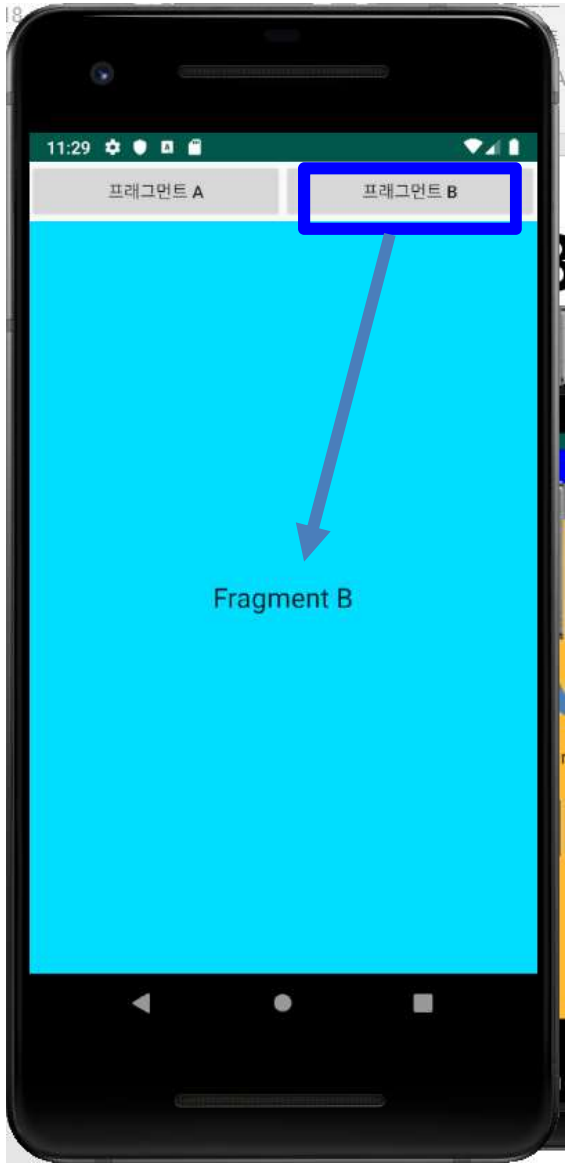
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/fragmentOne"
        android:name="com.ourincheon.ch9_project.FragmentA"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_a" />
    <fragment
        android:id="@+id/fragmentTwo"
        android:name="com.ourincheon.ch9_project.FragmentB"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        tools:layout="@layout/fragment_b" />
</LinearLayout>
```

소스코드 – 4쪽

실습 3: Fragment 동적 교체





코드에서 Fragment를 추가

1. Fragment 클래스의 객체를 생성하고, intent 인자를 전달

```
val fr: Fragment = FragmentA ()
fr.arguments = intent.extras
```

2. **FragmentManager** 객체를 생성한 다음,
   이 객체의 **beginTransaction** 메소드를 호출하여
   FragmentTransaction 객체를 반환 받음.

```
val transaction =
    supportFragmentManager.beginTransaction()
```

3. Activity 레이아웃에 FragmentA 에서 정의한 fragment 를 추가
   변경된 결과를 반영하려면 반드시 commit 메소드를 호출해야 함.

```
transaction.add(R.id.fragment_container, fr)
transaction.commit()
```

실습 3: setup

- Fragment 레이아웃
  - **fragment_a.xml, fragment_b.xml**
- Fragment 클래스 만들기
  - **FragmentA.kt**, **FragmentB.kt**
- Fragment를 Activity에 추가
  - **activity_main.xml**
- **Activity 작성**
  - **MainActivity.kt**
  - **FragmentManager** 객체를 사용하여 fragment 선택

실습 3: Activity 레이아웃에 container 추가

res/layout/activity_main.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/re
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout...>

    <LinearLayout...>

</LinearLayout>
```

소스코드 – 5쪽

```
<LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <Button
        android:text="프래그먼트 A"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:onClick="selectFragment"
        android:id="@+id/button1"
        android:layout_weight="1"/>

    <Button
        android:text="프래그먼트 B"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:onClick="selectFragment"
        android:id="@+id/button2"
        android:layout_weight="1"/>
</LinearLayout>
```

Fragment container

<fragment> 태그 대신
<LinearLayout> 태그를 사용

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

실습 3: Activity (1/2)

MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    private var fr:Fragment = FragmentA()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        if (fragment_container != null) {
            if (savedInstanceState != null)
                return
            val transaction =
                supportFragmentManager.beginTransaction()
            transaction.add(R.id.fragment_container, fr)
            transaction.commit()
        }
    }
```

이전에 저장된 fragment가 있으면, fragment를 추가할 필요가 없음

실습 3: Activity (2/2)

```
fun selectFragment(view: View) {
    fr = FragmentA()
    if (view.id == R.id.button2) {
        fr = FragmentB()
    }

    supportFragmentManager.beginTransaction()
        .replace(R.id.fragment_container, fr)
        .addToBackStack (null)
        .commit()
}
```

1. 어느 fragment를 선택했는가?(어떤 버튼을 눌렀는가?)
2. 선택된 fragment의 layout으로 fragment container를 교체(replace)한다.
   (FragmentManager 객체에서 replace 트랜잭션을 실행)
3. 사용자가 back 버튼(또는 취소 버튼)을 누르는 상황을 반영한다.
   이전 fragment를 back stack에 저장.
   back 버튼을 누르면 이전 fragment를 복원

잠깐! coding style

```
val transaction = supportFragmentManager.beginTransaction()
transaction.replace(R.id.fragment_container, fr)
transaction.addToBackStack (null)
transaction.commit()
```

```
supportFragmentManager.beginTransaction()
    .replace(R.id.fragment_container, fr)
    .addToBackStack(null)
    .commit()
```

Fragment 간 통신 구현

- fragment간, fragment와 activity간 통신은 어떻게 이루어질까?
  - Fragment끼리 직접 통신할 수 없음.
  - 모든 통신은 (hosting) **Activity**를 거쳐 이루어짐.

  - **Activity → fragment**
    - **findViewById** 메소드 ➔ fragment 객체의 **id** 참조
  - **Fragment → Activity**
    - **Fragment**
      - listener 인터페이스 정의 : **callback 메소드 선언**
      - onAttach 메소드 재정의 : Hosting activity에 대한 참조를 얻음
      - Activity 통신이 필요한 경우
        - » Hosting activity의 **callback 메소드 호출**
    - **Hosting Activity**
      - Listener 인터페이스 구현 상속 : **Callback 메소드 구현**

Fragment → Activity (1/2)

1. listener 인터페이스 선언

Fragment

```
class ToolbarFragment : Fragment ( ) {
    ToolbarListener activityCallback

    interface ToolbarListener {
            fun onButtonClick(position: Int, text: String)
    }
}
```

2. onAttach 메소드 오버라이딩
→ fragment를 포함하는 hosting activity를 참조
→ hosting activity가 listener 인터페이스를 제대로 구현했는지 확인해야 함. (그렇지 않으면 실행 중단!)

```
Override fun onAttach (context: Context) {
    activityCallback = context as ToolbarListener
}
```

3. fragment에 포함된 view에 대한 이벤트 처리
→ hosting activity의 callback 메소드 호출

```
private fun buttonClicked (view: View) {
    activityCallback . onButtonClicked (arg1, arg2)
}
```

```
button?.setOnClickListener { v: View -> buttonClicked(v) }
```

Fragment → Activity (2/2)

listener 인터페이스 구현

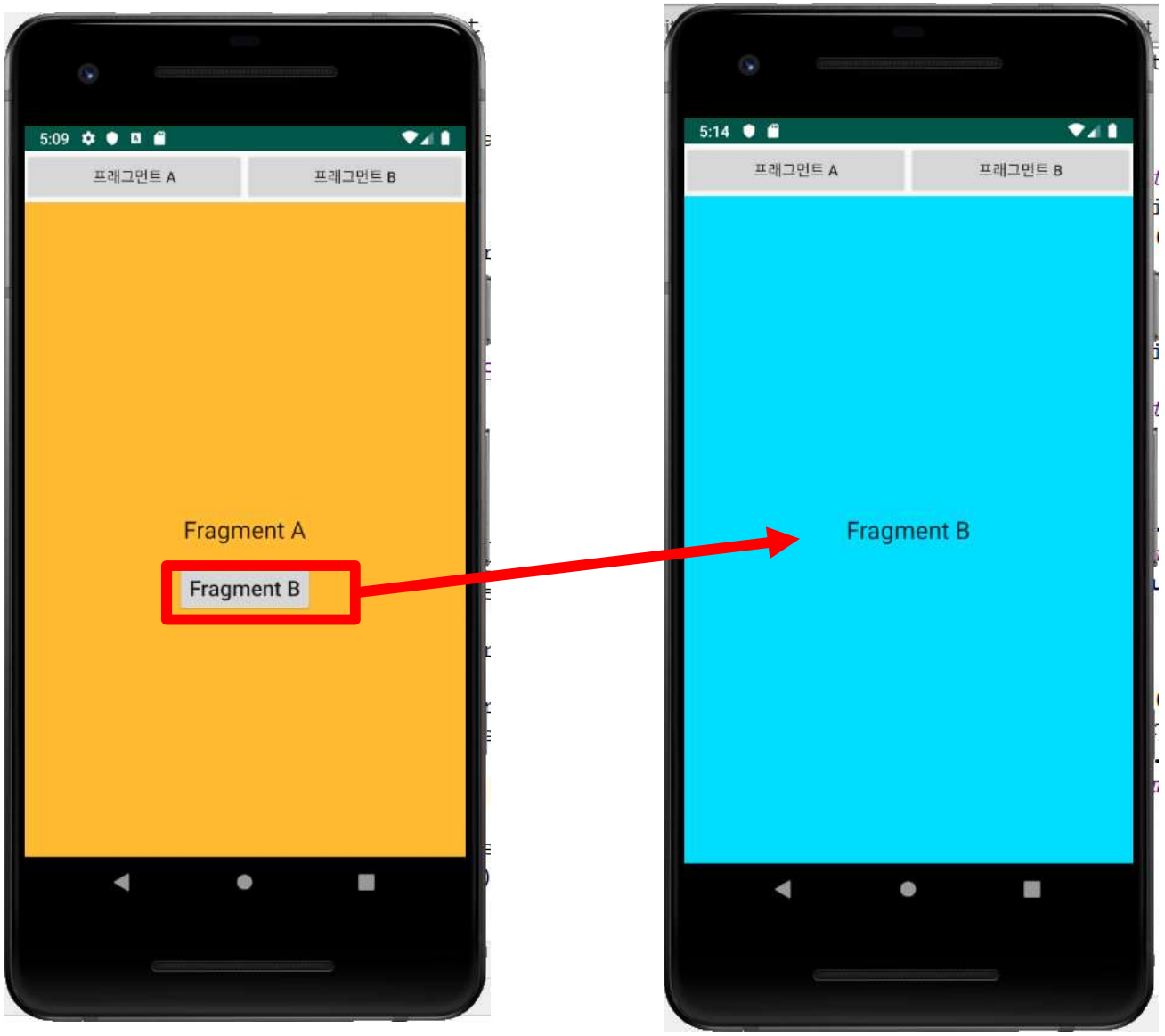
Hosting activity

```
class MainActivity  : FragmentActivity( ),
                      ToolbarFragment. ToolbarListener {

    . . . . . .

    override fin onButtonClick (arg1: Int, arg2: String) {
            // callback 메소드 구현
            . . . . . .
    }
}
```

실습 4: Fragment간 통신



실습 4: setup

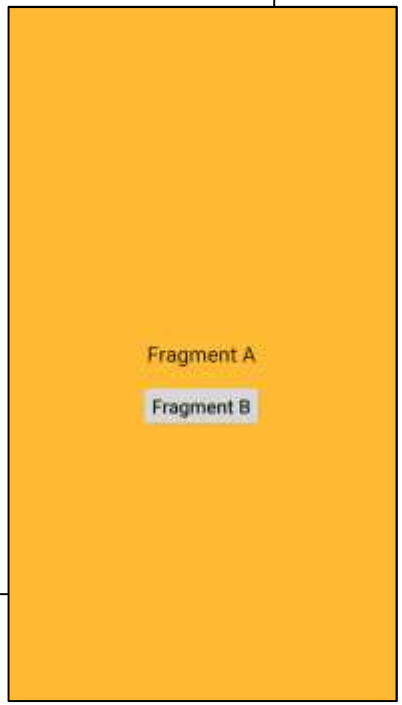
- Listener 인터페이스를 사용하여 구현
  - 실습 3의 FragmentA 에 button 1개 추가
    - button 클릭 ➔ FragmentB 로 전환
- Fragment 레이아웃
  - **fragment_a.xml (수정), fragment_b.xml**
- **Fragment 클래스 만들기**
  - **FragmentA.kt ➔ Activity의 메소드 호출 (수정)**
  - **FragmentB.kt**
- Fragment를 Activity에 추가
  - **activity_main.xml**
- **Activity 작성**
  - **MainActivity.kt (수정) - Fragment에서 요청한 메소드 실행**

실습 4: FragmentA 레이아웃에 button 추가

res/**layout**/**fragment_a.xml**

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_light">

    <TextView...>
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16sp"
        android:textSize="20sp"
        android:textAllCaps="false"
        android:text="Fragment B" />
</RelativeLayout>
```



소스코드 – 7쪽

실습 4: FragmentA

FragmentA.kt

```
override fun onCreateView(inflater: LayoutInflater,
                         container: ViewGroup?,
                         savedInstanceState: Bundle?): View? {
    val rootView = inflater.inflate(R.layout.fragment_a,
                        container, false)
    val button: Button? = rootView?.findViewById(R.id.button)
    button?.setOnClickListener {
        buttonClicked(it)
    }
    return rootView
}


override fun onAttach(context: Context?) {
    super.onAttach(context)
    try {
        activityCallback = context as ButtonListener
    } catch (e:ClassCastException) {
        throw ClassCastException(context?.toString()
                + " must implement buttonListener")
    }
}

private fun buttonClicked(view:View) {
    activityCallback?.onButtonClick()
}
```

```
var activityCallback: ButtonListener? = null

interface ButtonListener {
    fun onButtonClick()
}
```

소스코드 – 7~8쪽

실습 4: Activity

MainActivity.kt

```
class MainActivity : AppCompatActivity(),
                     FragmentA.ButtonListener {

    override fun onCreate(savedInstanceState: Bundle?) {...}

    fun selectFragment(view: View) {...}

    override fun onButtonClick() {
        supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, FragmentB())
            .commit()
    }
}
```

소스코드 – 8~9쪽

실습 5: fragment 간 통신

실습 5: 준비

- Fragment 레이아웃
  - **toolbar_fragment.xml, text_fragment.xml**
- Fragment 클래스 만들기
  - **ToolbarFragment.kt**, **TextFragment.java**
    - **ToolbarFragment 에서 입력한 문자열을 Seekbar에서 정의한 text size로 TextFragment에 출력**
- Fragment를 Activity에 추가
  - **activity_main.xml**
- Activity 작성
  - **MainActivity.kt**

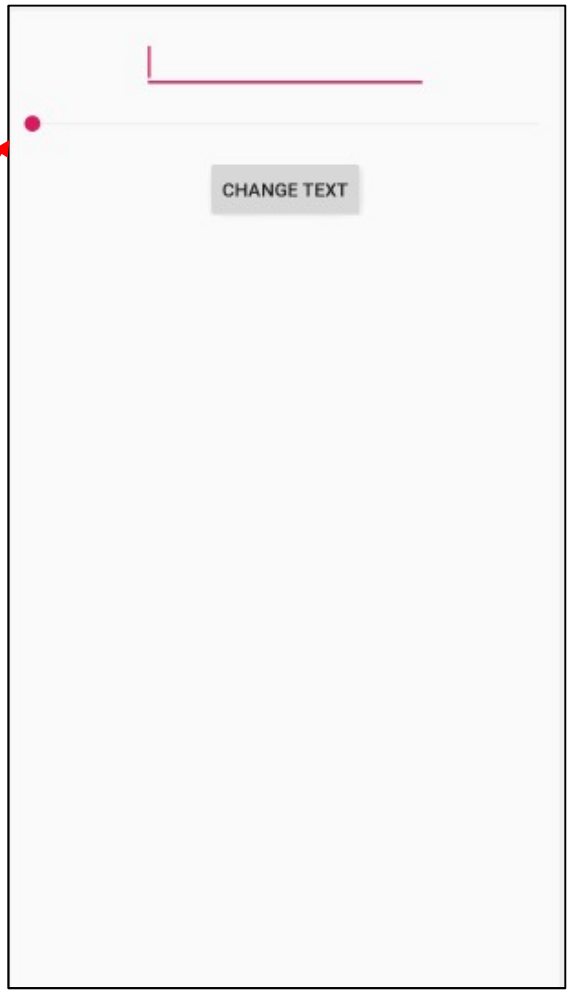
실습 5: toolbar_fragment.xml - Layout

res/layout/toolbar_fragment.xml

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/seekBar1"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="16dp"
    android:text="@string/change_text" />

<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="16dp"
    android:ems="10"
    android:inputType="text"
    android:importantForAutofill="no">
    <requestFocus />
</EditText>

<SeekBar
    android:id="@+id/seekBar1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/editText1"
    android:layout_marginTop="14dp"/>
```



소스코드 – 10쪽

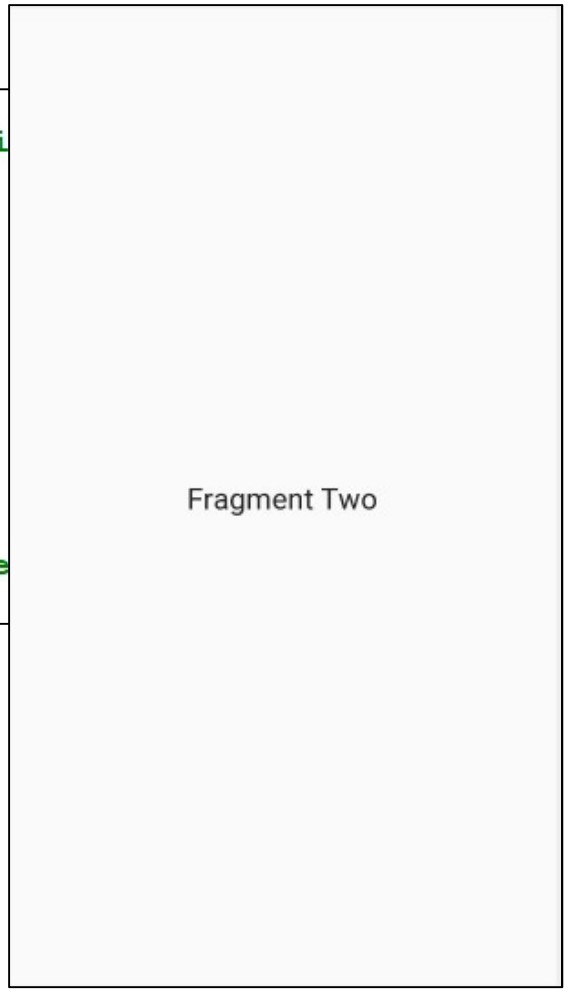
실습 5: text_fragment.xml - Layout

res/layout/text_fragment.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fragment_two"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:textAppearance="?android:attr/textAppearance
</RelativeLayout>
```

소스코드 – 10~11쪽



실습 5: activity_main.xml - Layout

res/layout/activity_main.xml

```
<fragment
    android:id="@+id/toolbar_fragment"
    android:name="edu.incheon.ch9project.ToolbarFragment"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:layout="@layout/toolbar_fragment" />

<fragment
    android:id="@+id/text_fragment"
    android:name="edu.incheon.ch9project.TextFragment"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/toolbar_frag
    tools:layout="@layout/text_fragment" />
```

소스코드 – 11쪽

실습 5: ToolbarFragment.kt

```
class ToolbarFragment : Fragment(), SeekBar.OnSeekBarChangeListener {

    var seekvalue = 10

    var activityCallback: ToolbarListener? = null

    interface ToolbarListener {
        fun onButtonClick(position: Int, text: String)
    }

    override fun onAttach(context: Context) {...}

    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {...}

    private fun buttonClicked(view: View) {
        activityCallback?.onButtonClick(seekvalue,
            editText1.text.toString())
    }

    override fun onProgressChanged(seekBar: SeekBar?, progress: Int,
                                   fromUser: Boolean) {...}

    override fun onStartTrackingTouch(seekBar: SeekBar?) {...}

    override fun onStopTrackingTouch(seekBar: SeekBar?) {...}
}
```

실습 5: TextFragment.kt

```
class TextFragment() : Fragment() {

    override fun onCreateView(inflater: LayoutInflater,
                              container: ViewGroup?,
                              savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.text_fragment,
            container, false)
    }

    fun changeTextProperties(fontsize: Int, text: String) {
        textView1.textSize = fontsize.toFloat()
        textView1.text = text
    }

}
```

실습 5: MainActivity.kt

```
class MainActivity : AppCompatActivity(),
    ToolbarFragment.ToolbarListener {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    override fun onButtonClick(fontsize: Int, text: String) {
        val textFragment = supportFragmentManager.findFragmentById(
            R.id.text_fragment) as TextFragment

        textFragment.changeTextProperties(fontsize, text)
    }
}
```