

소프트웨어설계및실험

2024 Spring

Jetpack Library

이론

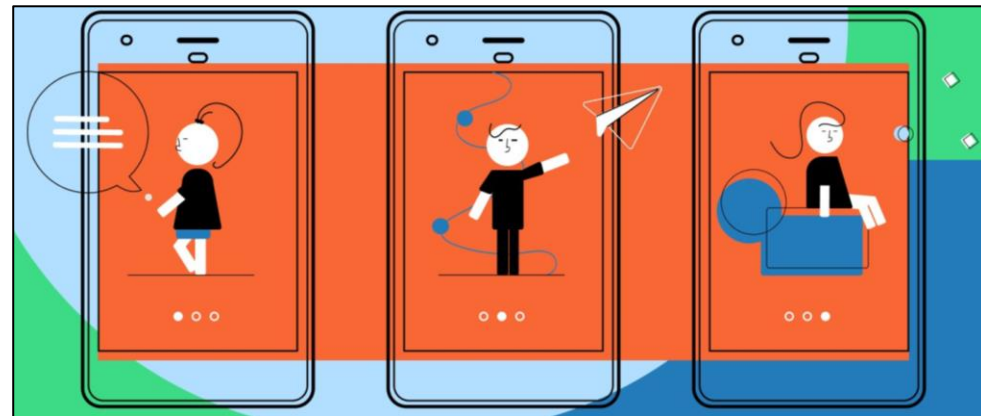
- ◆ Jetpack 과 androidx
 - ◆ 라이브러리 개요
 - ◆ Appcompat 라이브러리
- ◆ Androidx 활용
 - ◆ Fragment
 - ◆ viewPager
 - ◆ Drawer layout
 - ◆ recyclerview

❖ Jetpack 이란?

- 구글에서 개발된 Android 앱 개발용 확장 라이브러리
- Androidx로 시작하는 패키지명을 사용한다.
- 사용 목적
 - 앱을 개발하는데 필요한 권장 아키텍처를 제공한다.
 - 플랫폼 API에서 제공하지 않는 다양한 기능을 제공한다.
(Ex.) 스와이프를 통한 뷰 변경)
 - 플랫폼 API의 호환성 문제를 해결 가능하다.
(Ex.) API 21 버전 이하는 Toolbar widget 사용이 불가)

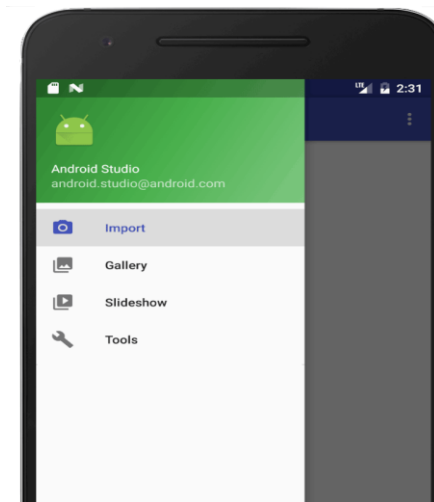
❖ 화면 구성과 관련된 androidx 라이브러리

- **androidx.appcompat** : 앱 API 레벨 호환성 해결
- **androidx.fragment** : 액티비티처럼 동작하는 뷰
- **androidx.recyclerview** : 목록 화면 구성
- **androidx.viewpager2** : 스와이프로 넘기는 화면 구성
- **androidx.drawerlayout** : 옆에서 서랍처럼 열리는 화면 구성
- 그 외 라이브러리 :
<https://developer.android.com/jetpack/androidx/explorer?hl=ko>



<androidx.viewpager : 스와이프를 통한 화면 전환>

medium.com/proandroiddev/viewpager2-and-diffutil-d853cdab5f4a

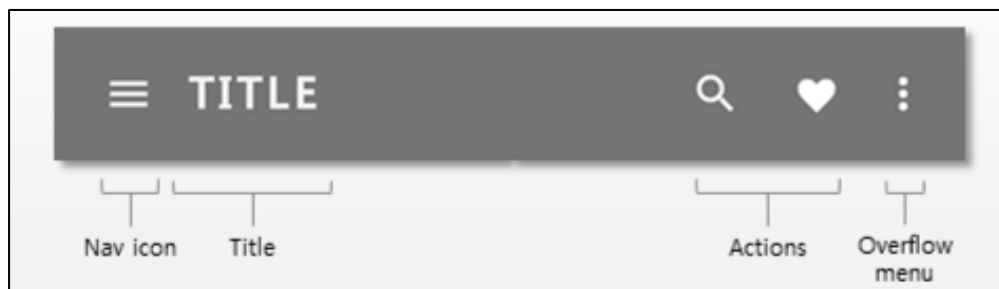


<androidx.drawerlayout : 스와이프를 통한 드로어 열기>

stackoverflow.com/questions/59853488/navigation-drawer-get-the-new-layout

❖ Appcompat?

- 기존 플랫폼 API와 동일하게 Android 앱의 화면을 구성하는 액티비티를 만들 수 있다.
- API 레벨마다 다르게 제공되는 widget을 통합하여 사용 가능하다.(호환성 해결)
- API 호환 해결 외에도 앱 테마, 액티비티 상단의 액션바를 다룰 수 있다.



<actionbar의 구성> recipes4dev.tistory.com

❖ Appcompat 활용 – API 호환성 해결

- Gradle 파일의 dependency 항목에 appcompat 추가
 - implementation 'androidx.appcompat:appcompat:1.2.0'
- Activity 파일의 appcompat 라이브러리 추가
 - import androidx.appcompat.app.AppCompatActivity
- AppCompatActivity 클래스를 상속받아 작성
 - class MainActivity : AppCompatActivity() {

기존 호환되지 않는 위젯(버튼) 선언

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```



appcompat을 통해 호환성이 지원되는 위젯(버튼) 선언

```
<androidx.appcompat.widget.AppCompatButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
```

❖ Fragment?

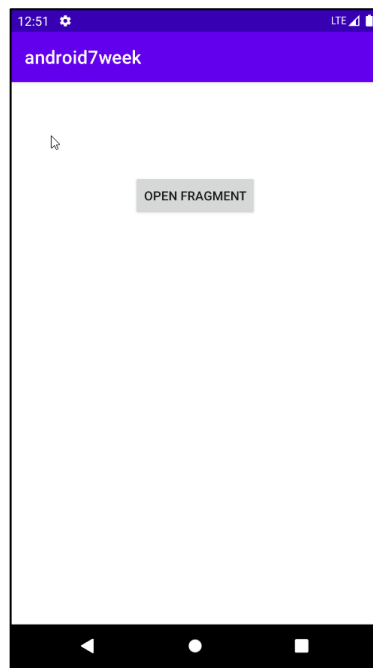
- 액티비티처럼 동작 가능하지만, **한 화면 내에 여러 뷰로 동작 가능**하다.
- 액티비티 클래스 구현의 복잡성을 해결하기 위해 사용된다.

❖ Fragment 구현 방법

1. **Fragment 클래스를 선언하고 원하는 동작을 액티비티와 동일하게 View 형태로 구현**
2. **Fragment를 출력할 뷰, 레이아웃 생성**
3. Fragment 동적 제어(추가, 제어 등)를 위해 **FragmentManager로 만든 FragmentTransaction 객체를 선언**
4. 액티비티 제어 코드내에서 `transaction.add().commit()` 메소드를 호출하여 객체를 화면에 출력

```
class OneFragment : Fragment() {  
    override fun onCreateView( ① View로 생성되는 Fragment  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? { ② Fragment 출력 Layout  
        return inflater.inflate(R.layout.fragment_bind, container, attachToRoot: false)  
    }  
}
```

Fragment 클래스 정의



Fragment 출력 화면

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"  
    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">  
    <androidx.appcompat.widget.AppCompatButton...>  
  
    <FrameLayout ② Fragment 출력 뷰  
        android:id="@+id/fragment_content"  
        android:layout_width="match_parent"  
        android:layout_height="600px"  
        android:layout_marginTop="50px"  
        app:layout_constraintBottom_toBottomOf="parent"  
        app:layout_constraintLeft_toLeftOf="parent"  
        app:layout_constraintRight_toRightOf="parent"  
        app:layout_constraintTop_toBottomOf="@+id/fragBut1" />  
    </androidx.constraintlayout.widget.ConstraintLayout>
```

메인 액티비티 레이아웃

❖ Fragment?

- 액티비티처럼 동작 가능하지만, **한 화면 내에 여러 뷰로 동작 가능**하다.
- 액티비티 클래스 구현의 복잡성을 해결하기 위해 사용된다.

❖ Fragment 구현 방법

1. Fragment 클래스를 선언하고 원하는 동작을 액티비티와 동일하게 View 형태로 구현
2. Fragment를 출력할 뷰, 레이아웃 생성
3. **Fragment 동적 제어(추가, 제거 등)를 위해 FragmentManager로 만든 FragmentTransaction 객체를 선언**
4. **액티비티 제어 코드내에서 transaction.add().commit() 메소드를 호출하여 객체를 화면에 출력**

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val fragButton = findViewById<Button>(R.id.fragBut1)  
        val fragmentManager: FragmentManager = supportFragmentManager  
        var onClicked = false  
        fragButton.setOnClickListener { it: View!  
            if (onClicked) {  
                onClicked = false  
                val transaction: FragmentTransaction = fragmentManager.beginTransaction()  
                val frameLayout = supportFragmentManager.findFragmentById(R.id.fragment_content)  
                transaction.remove(frameLayout!!).commit()  
            }  
            else {  
                onClicked=true  
                val transaction: FragmentTransaction = fragmentManager.beginTransaction()  
                transaction.add(R.id.fragment_content, OneFragment()).commit()  
            }  
        }  
    }  
}
```

③ Fragment 제어 객체 선언

③ Fragment 트랜잭션 객체 선언

④ Fragment 제거

④ Fragment 추가

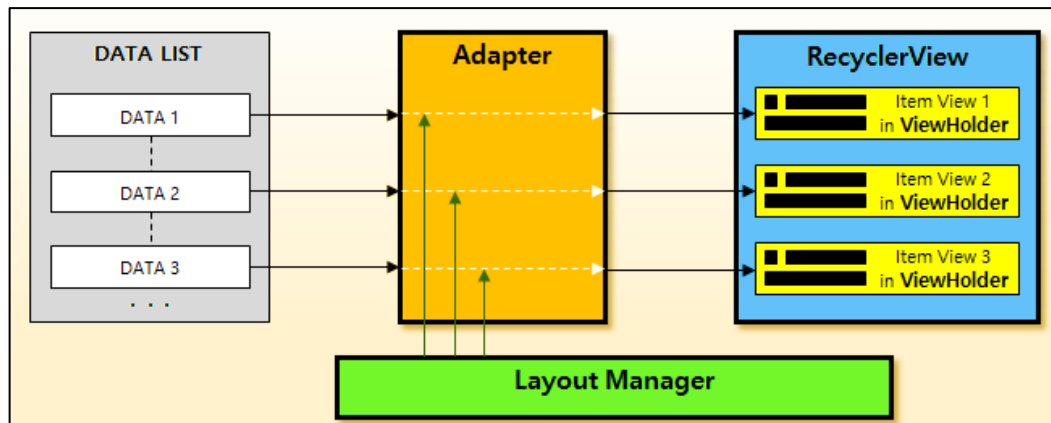
<메인 액티비티 클래스 정의>

❖ 리사이클러 뷰?

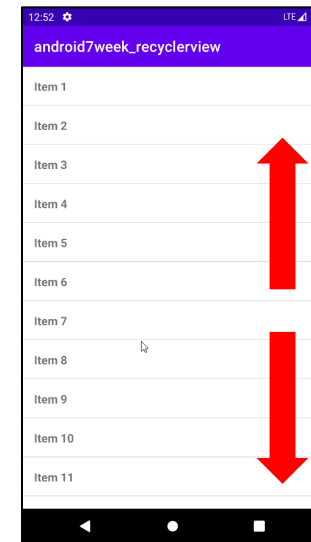
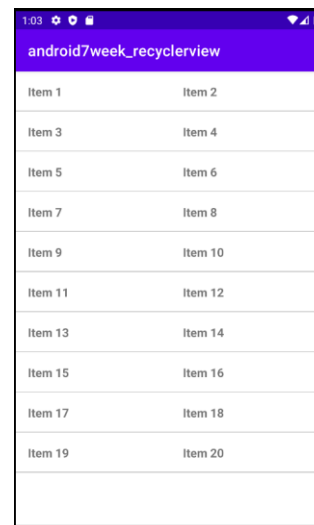
- 여러 가지 항목을 리스트로 나열 시켜주는 뷰
- 플랫폼 API의 리스트 뷰보다 많은 기능을 제공

❖ 리사이클러뷰 구성 요소

- **ViewHolder** : 항목에 필요한 뷰 객체를 가진다.
- **Adapter** : 항목을 구성하는 역할을 담당한다.
- **LayoutManager** : 항목을 배치하고 리스트 형태의 출력을 담당한다.
- (옵션) **ItemDecoration** : 항목을 꾸민다.



recipes4dev.tistory.com



위아래 스크롤을 통한
목록 확인

리사이클러뷰 출력 화면(좌: 그리드, 우: 선형)

```
11 class RecyclerViewActivity : AppCompatActivity() {
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         val binding = RecyclerViewBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16         val datas = mutableListOf<String>()
17         for(i in 1..20){datas.add("Item $i")}
18
19         binding.recyclerView.layoutManager = LinearLayoutManager(context: this)
20         binding.recyclerView.adapter = MyAdapter(datas)
21         binding.recyclerView.addItemDecoration(DividerItemDecoration(
22             context: this, LinearLayoutManager.VERTICAL))
23     }
24 }
25 }
```

리사이클러뷰 각 구성 요소 선언

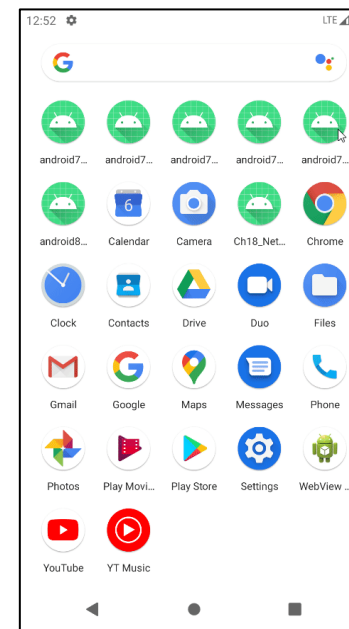
<리사이클러뷰 액티비티 클래스>

❖ 뷰페이저?

- **스와이프 이벤트로 화면을 전환**하는 기능
(Orientation 조절 가능)

❖ 뷰페이저의 2가지 구현 방식

- **RecyclerView.Adapter**를 이용한 뷰 체인지
 1. 기존 리사이클러뷰의 어댑터를 viewPager.adapter로 변경하여 출력
- **FragmentStateAdapter**를 이용한 뷰 체인지
 1. Fragment를 담고 있는 리스트 선언
 2. createFragment() 함수에서 반환하는 Fragment 객체를 출력



스와이프로 화면 전환

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        val binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
        val datas = mutableListOf<String>()  
        for(i in 1..3){datas.add("Item $i")}  
        binding.viewpager.adapter = MyPagerAdapter(datas)  
        binding.viewpager.orientation = ViewPager2.ORIENTATION_HORIZONTAL  
    }  
}
```

리사이클러뷰 어댑터

```
class MyFragmentPagerAdapter(activity: FragmentActivity): FragmentStateAdapter(activity){  
    val fragments: List<Fragment>  
    init{  
        fragments= listOf(OneFragment(), TwoFragment(), ThreeFragment())  
    }  
}
```

Fragment 어댑터

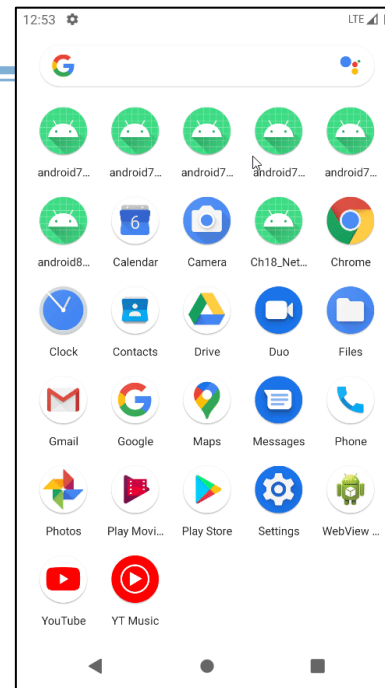
<뷰페이저의 2가지 구현 방법>

❖ 드로어 레이아웃?

- 액티비티 화면에 **보이지 않던 내용이 스와이프를 통해 밀려 나오는 기능**
- androidx의 라이브러리인 DrawerLayout은 마치 서랍처럼 열리는 메뉴를 구성

❖ 드로어 레이아웃 구현 방법

1. DrawerLayout 태그를 최상위로 가지는 레이아웃 할당
2. 최상위 태그 내에 2개 이상의 하위 태그 선언
(각각 MainLayout, DrawerLayout 뷰를 담당)
3. 액티비티에 토글 객체 선언 및 드로어 제어



DrawerLayout 출력

1. DrawerLayout 최상위 태그

```
<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawer">
    <LinearLayout...>
    <TextView...>
</androidx.drawerlayout.widget.DrawerLayout>
```

<드로어 레이아웃 선언>

```
override fun onCreate(savedInstanceState: Bundle?) {
    var binding = ActivityMainBinding.inflate(layoutInflater)
    super.onCreate(savedInstanceState)
    setContentView(binding.root)

    toggle = ActionBarDrawerToggle(activity, this, binding.drawer, R.string.drawer_open, R.string.drawer_close)
    toggle.syncState()

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        if (toggle.onOptionsItemSelected(item)) {return true}
        return super.onOptionsItemSelected(item)
    }
}
```

3. 토글 객체 선언

3. 드로어 제어

<토글 객체 선언 및 드로어 제어>

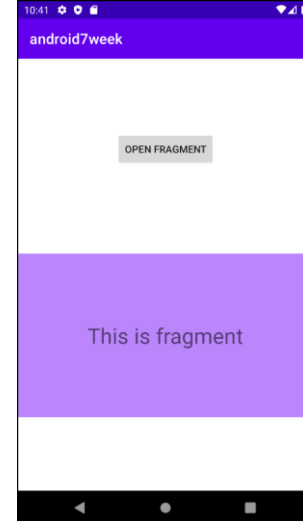
실습

- ◆ 실습 (따라하기)
 - ◆ 예제 1 - Fragment
 - ◆ 예제 2 - RecyclerView
 - ◆ 예제 3 - ViewPager
 - ◆ 예제 4 - drawerLayout

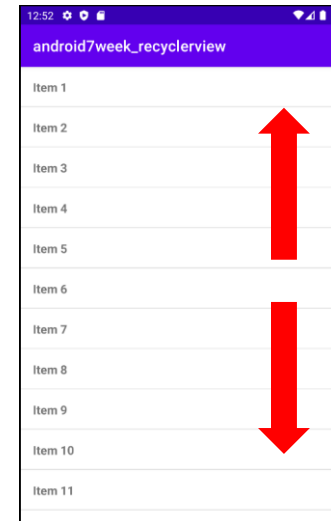
- ◆ 응용
 - ◆ 예제 5 - 응용

❖ Androidx를 통한 앱 개발

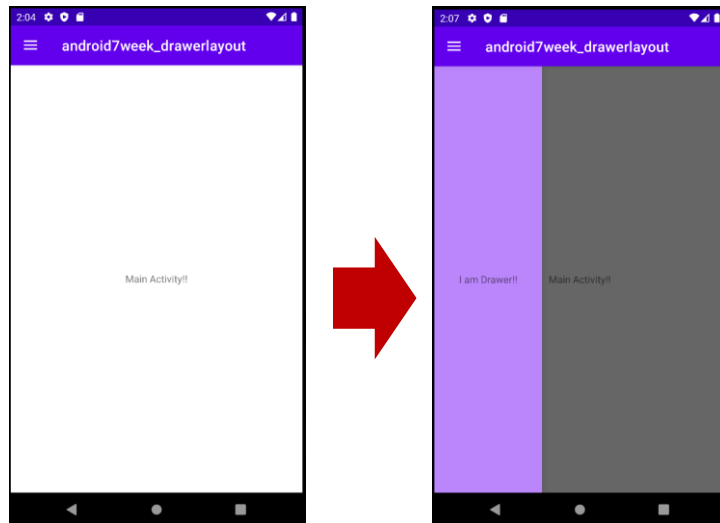
- Androidx에서 제공하는 주요 기능을 활용하여 어플리케이션을 개발한다.
1. Fragment [On/Off Switch]
 2. RecyclerView [Item List]
 3. ViewPager [Background Color Switch by swiping]
 4. DrawerLayout [Pull Out Drawer]



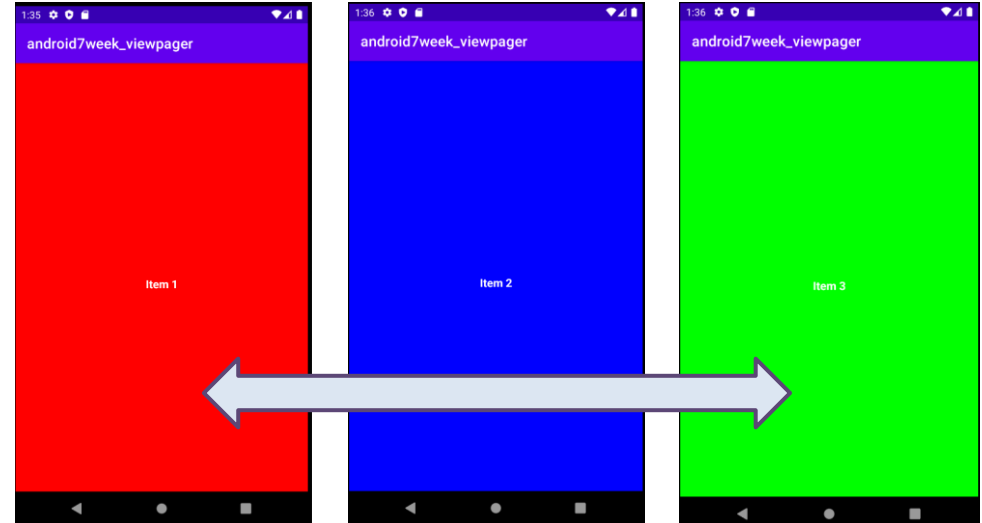
<Fragment>



<RecyclerView>



<DrawerLayout>

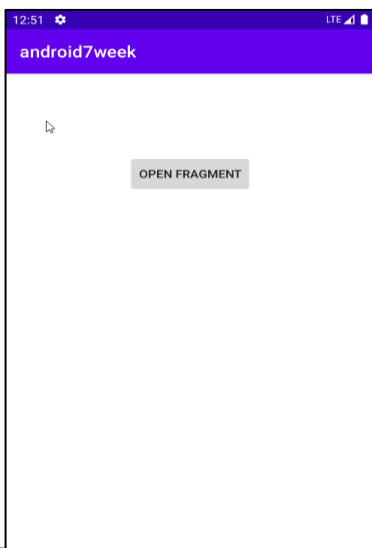


<ViewPager>

예제 1 – Fragment

❖ Fragment 객체 제어

- (16 line) : Fragment 동적 제어를 위한 객체 할당
- (21-23 line) : Fragment를 뷰에서 제거(Invisible)
- (27-28 line) : Fragment를 뷰에 할당(Visible)
- `supportFragmentManager` : Fragment 제어 매니저
- `Transaction.add().commit()` : Fragment 추가 및 적.



MainActivity.kt

```
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14
15         Button과 Manager 선언하기
16
17         var onClicked = false
18         fragButton.setOnClickListener { it: View!
19             if (onClicked) {
20                 onClicked = false
21                 Fragment 삭제하기
22             }
23             else {
24                 onClicked=true
25                 Fragment 추가하기
26             }
27         }
28     }
29 }
30
31
32 }
```

예제 1 – Fragment

❖ Fragment 클래스 구현

- (10 line) : Fragment는 뷰 단위로 동작
- (15 line) : 뷰에 layout(xml)을 적용

❖ Fragment 레이아웃 구현

아래와 같이 색상이 감지되지 않을 경우에는
values/colors.xml 에 색상값을 추가

```
android:orientation="vertical"
android:background="@color/purple_200">
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <color name="black">#FF000000</color>
4   <color name="white">#FFFFFFF</color>
5   <color name="purple_200">#FFBB86FC</color>
6 </resources>
```

OneFragment.kt

```
9 class OneFragment : Fragment() {
10     override fun onCreateView(
11         inflater: LayoutInflater,
12         container: ViewGroup?,
13         savedInstanceState: Bundle?
14     ): View? {
15         inflater.inflate(R.layout.fragment_one, container, false)
16     }
17 }
```

fragment_bind.xml

```
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     android:orientation="vertical"
7     android:background="@color/purple_200">
8
9     <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         app:layout_constraintTop_toTopOf="parent"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintRight_toRightOf="parent"
15         app:layout_constraintLeft_toLeftOf="parent"
16         android:text="This is fragment"
17         android:id="@+id/frag_text"
18         android:textSize="30dp"/>
19 </androidx.constraintlayout.widget.ConstraintLayout>
```

❖ 메인 레이아웃 내 Fragment 배치

- (8-16 line) : Fragment On/Off를 위한 appcompat 버튼
- (18-26 line) : Fragment가 위치할 FrameLayout
- **ConstraintLayout** : 위젯, 레이아웃 간 상대적 위치 지정
- **{Current}_to{Target}Of** : 대상의 {Target}에 {Current}가 위치하도록 지정

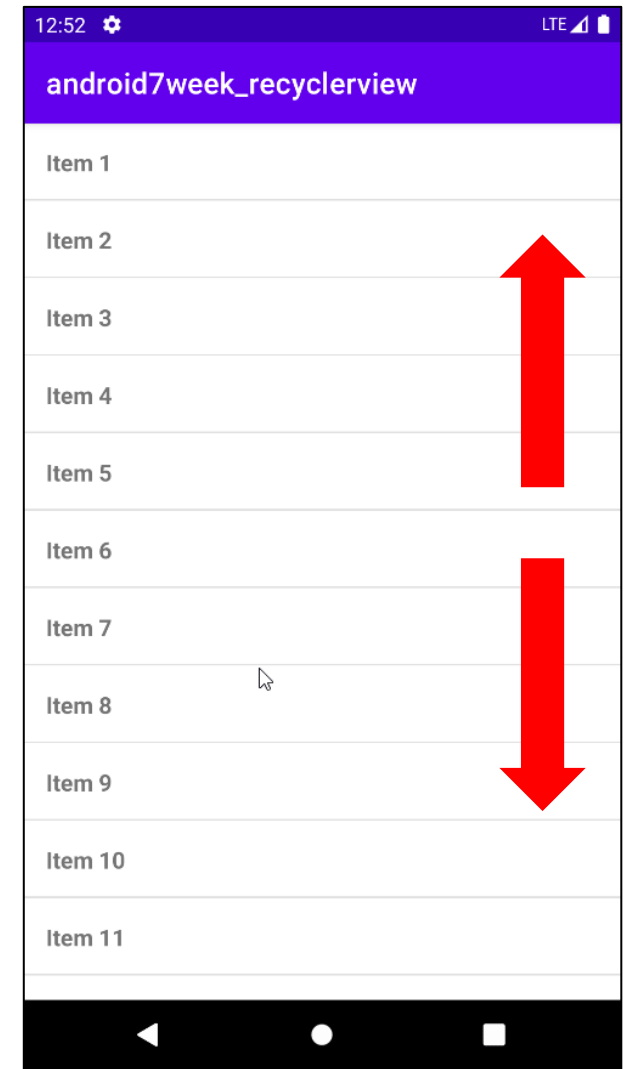
activity_main.xml

```
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8     <androidx.appcompat.widget.AppCompatButton
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:text="Open Fragment"
12        android:id="@+id/fragBut1"
13        app:layout_constraintBottom_toTopOf="@+id/fragment_content"
14        app:layout_constraintLeft_toLeftOf="parent"
15        app:layout_constraintRight_toRightOf="parent"
16        app:layout_constraintTop_toTopOf="parent"/>
17
18    <FrameLayout
19        android:id="@+id/fragment_content"
20        android:layout_width="match_parent"
21        android:layout_height="600px"
22        android:layout_marginTop="50px"
23        app:layout_constraintBottom_toBottomOf="parent"
24        app:layout_constraintLeft_toLeftOf="parent"
25        app:layout_constraintRight_toRightOf="parent"
26        app:layout_constraintTop_toBottomOf="@+id/fragBut1" />
27 </androidx.constraintlayout.widget.ConstraintLayout>
```

<activity_main>

❖ 리사이클러뷰 구현

1. 프로젝트 기본 요소 변경하기
2. 레이아웃 구성하기
3. 리사이클러뷰 제어하기
4. 리사이클러뷰 어댑터 구현하기

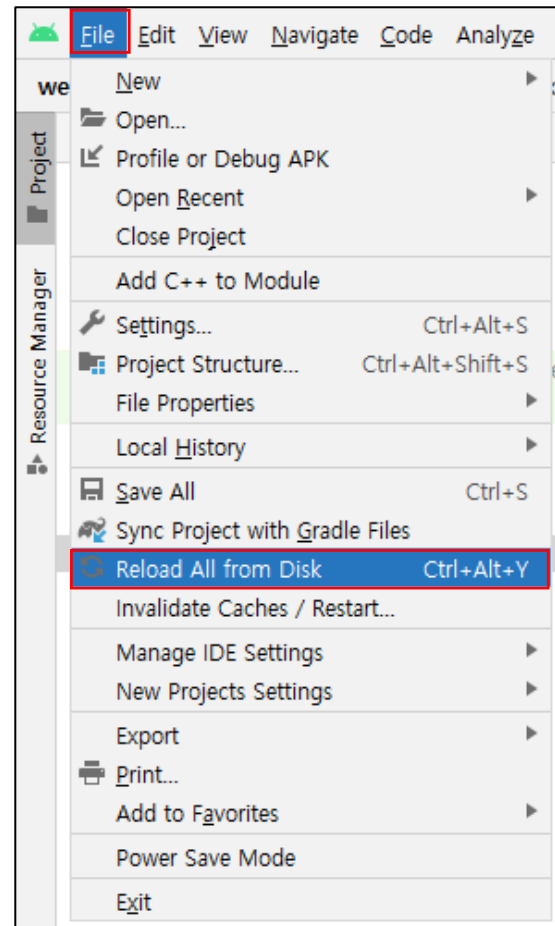
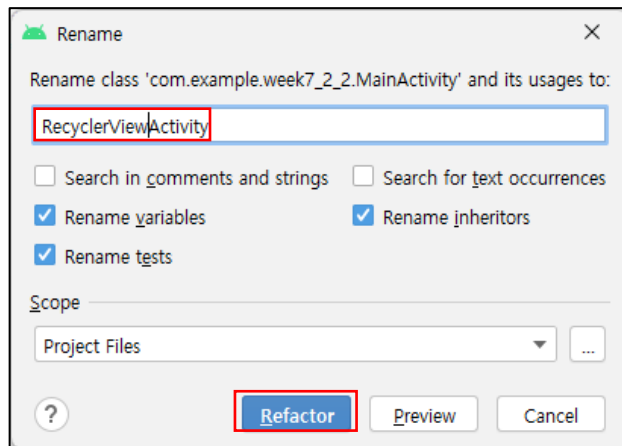
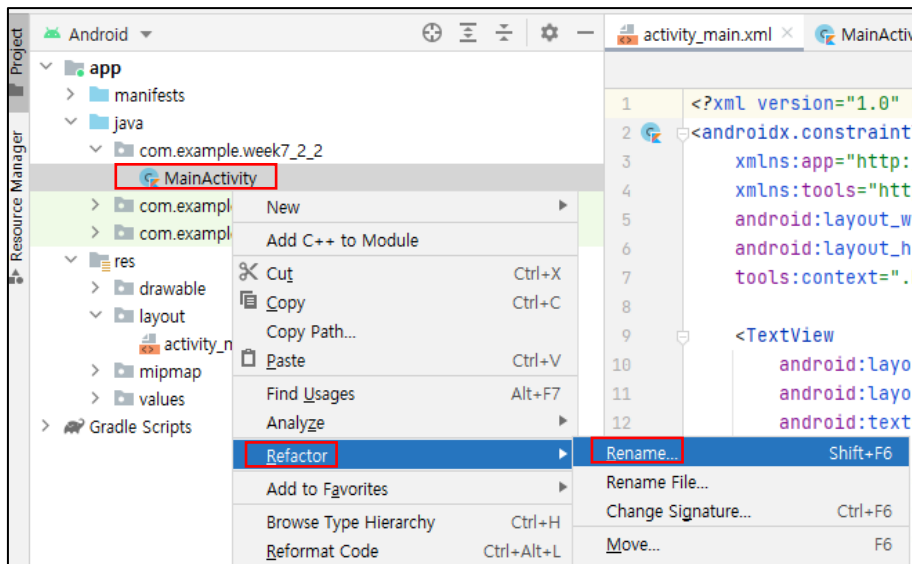


예제 2 – RecyclerView

1. 프로젝트 기본 요소 변경하기

- 이름 변경하기(Refactor – Rename)
 - MainActivity -> RecyclerViewActivity
 - activity_main.xml -> item_main.xml

프로젝트 동기화가 안될 경우만 수행

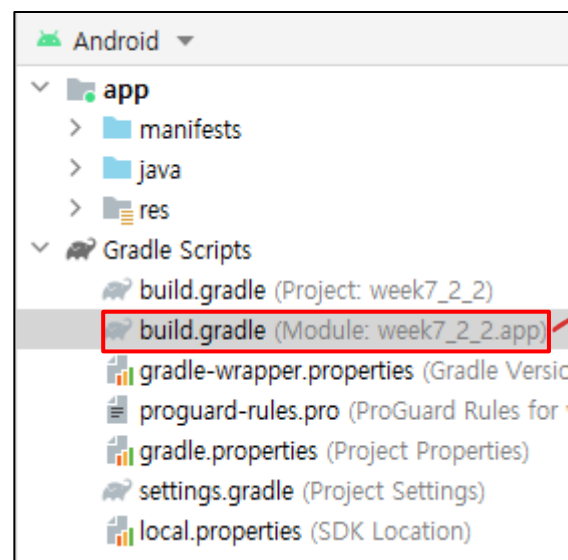


예제 2 – RecyclerView

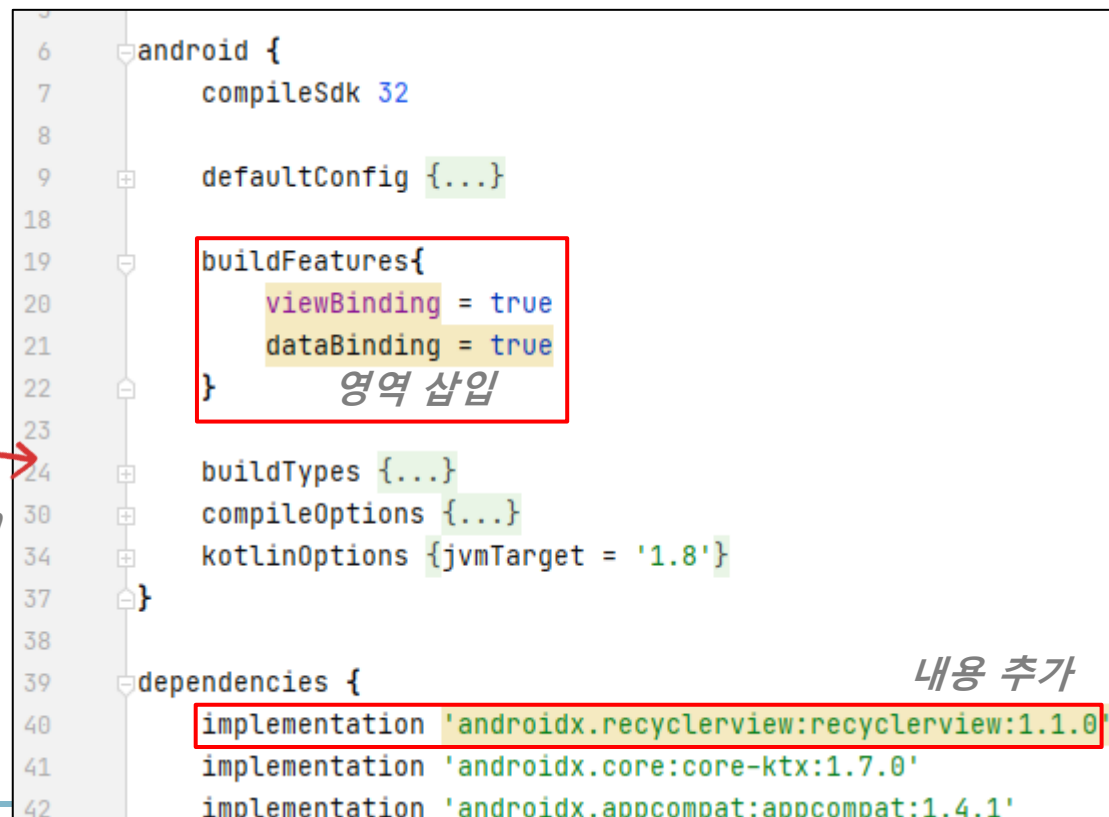
1. 프로젝트 기본 요소 변경하기

■ Build.gradle 추가하기

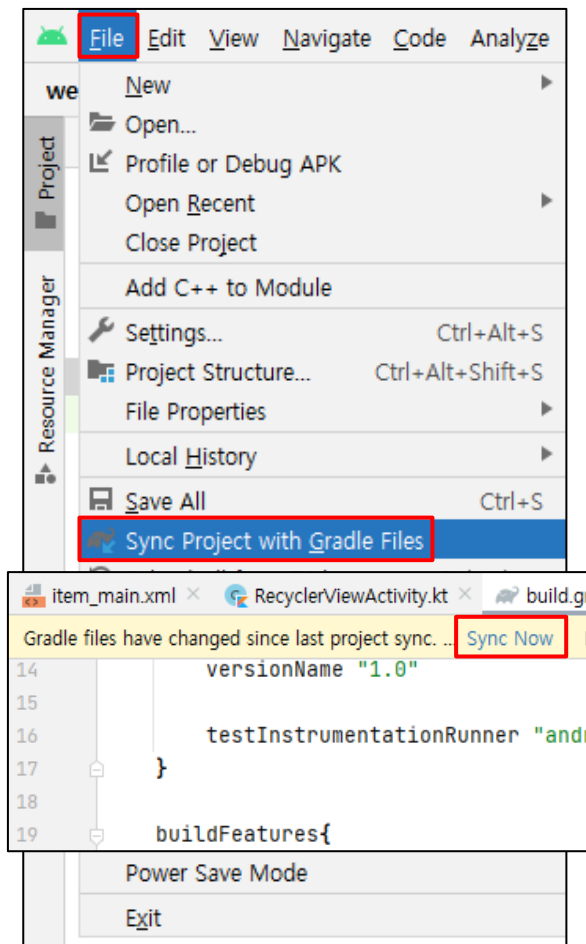
- 1) Build.gradle (Module: xxx.app) 열기
- 2) 추가하기
 - android {} 영역 내에 buildFeatures {} 영역 삽입
 - dependencies {} 영역 내에 implementation 내용 추가
- 3) 프로젝트 Sync 하기



open



Build.gradle(:app)



2. 레이아웃 구성하기

- Recycler_view.xml 작성

recycler_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/recyclerView"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</androidx.recyclerview.widget.RecyclerView>
```

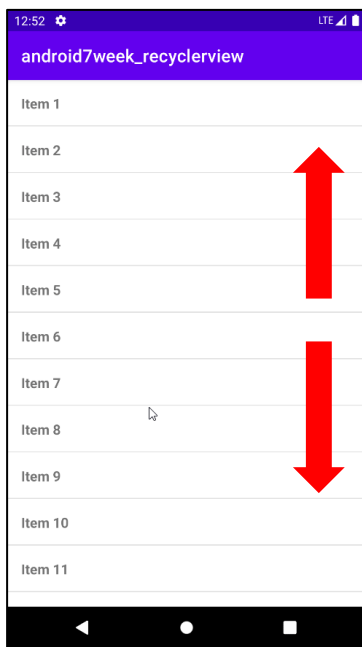
- Item_main.xml 작성

item_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/item_root"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp">
    <TextView
        android:id="@+id/item_data"
        android:layout_width="wrap_content"
        android:layout_height="20dp"
        android:textSize="16dp"
        android:textStyle="bold"/>
</LinearLayout>
```

3. 리사이클러뷰 제어하기

- (19-20 line) : Linear LayoutManager, adapter 할당
- (21 line) : 수직으로 리스트가 출력되기 위한 decoration



RecyclerViewActivity.kt

```
11 class RecyclerViewActivity : AppCompatActivity() {
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         val binding = RecyclerViewBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16         val datas = mutableListOf<String>()
17         for(i in 1..20){datas.add("Item $i")}
18
19
20
21
22
23
24     }
25 }
```

linearLayoutManager, adapter, itemDecoration 설정

4. 리사이클러뷰 어댑터 구현하기

- (12 line) : 리스트 원소 개수 반환
- (13-14 line) : 리사이클러뷰 객체 할당
- (16-23 line) : Adapter의 각 요소에 데이터 삽입

MyAdapter.kt

```
9      class MyViewHolder(val binding: ItemMainBinding): RecyclerView.ViewHolder(binding.root)
10
11     class MyAdapter(val datas:MutableList<String>): RecyclerView.Adapter<RecyclerView.ViewHolder>() {
12         override fun getItemCount(): Int = datas.size
13         override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder = MyViewHolder(
14             ItemMainBinding.inflate(LayoutInflater.from(parent.context), parent, attachToParent: false))
15
16         override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
17             Log.d( tag: "kkang", msg: "onBindViewHolder : $position")
18             val binding = (holder as MyViewHolder).binding
19
20             binding.itemData.text = datas[position]
21             binding.itemRoot.setOnClickListener{ it: View!
22                 Log.d( tag: "kkang", msg: "item root click : $position")
23             }
24         }
25     }
```

예제 3 – ViewPager

❖ 뷰페이지 구현

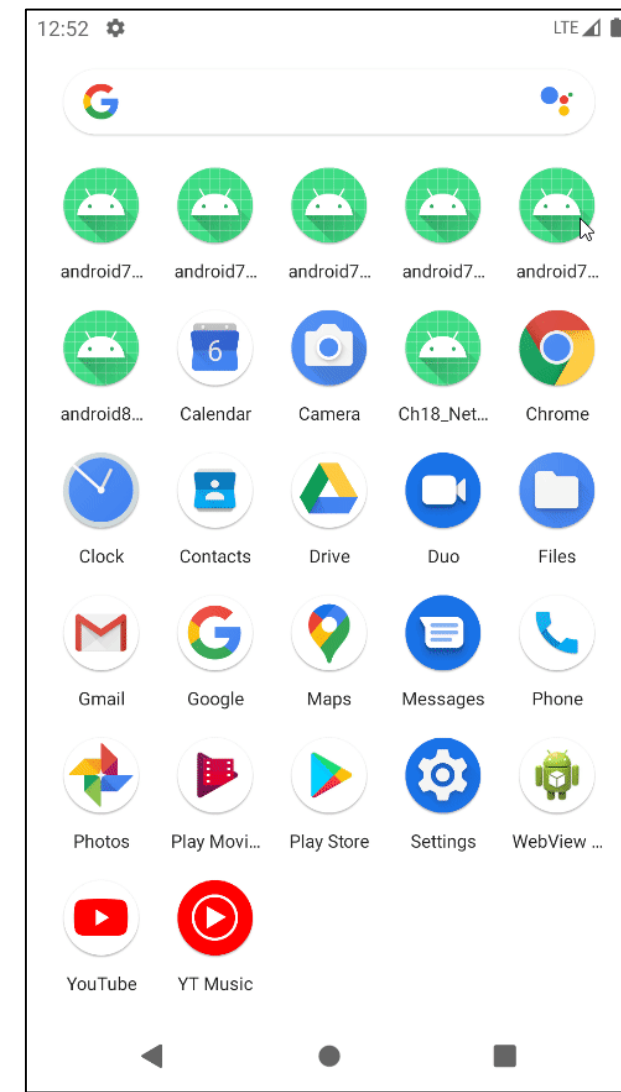
1. 레이아웃 구성하기
2. 뷰페이지 제어하기
3. 리사이클러뷰 활용하기

Build.gradle(:app)

```
6 android {
7     compileSdk 32
8
9     defaultConfig {...}
10
11     buildFeatures{
12         viewBinding = true
13         dataBinding = true
14     }
15
16     buildTypes {...}
17     compileOptions {...}
18     kotlinOptions {jvmTarget = '1.8'}
19 }
20
21 dependencies {
22     implementation 'androidx.recyclerview:recyclerview:1.1.0'
23     implementation 'androidx.core:core-ktx:1.7.0'
24     implementation 'androidx.appcompat:appcompat:1.4.1'
```

영역 삽입

내용 추가



1. 레이아웃 구성하기

- 메인 레이아웃
 - (2 line) : ViewPager2 위젯을 뷰 전체로 할당
- 뷰페이지저 레이아웃

activity_main.xml

```
2 <androidx.viewpager2.widget.ViewPager2
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:id="@+id/viewpager"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     >
8 </androidx.viewpager2.widget.ViewPager2>
```

item_pager.xml

```
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:id="@+id/item_pager"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     app:layout_constraintTop_toTopOf="parent"
9     app:layout_constraintBottom_toBottomOf="parent"
10    app:layout_constraintLeft_toLeftOf="parent"
11    app:layout_constraintRight_toRightOf="parent">
12
13    <TextView
14        android:id="@+id/item_pager_text_view"
15        android:layout_width="match_parent"
16        android:layout_height="match_parent"
17        android:textAlignment="center"
18        android:paddingTop="300dp"
19        android:textColor="@color/white"
20        android:textSize="16sp"
21        android:textStyle="bold"
22        app:layout_constraintBottom_toBottomOf="parent"
23        app:layout_constraintLeft_toLeftOf="parent"
24        app:layout_constraintRight_toRightOf="parent"
25        app:layout_constraintTop_toTopOf="parent" />
26 </androidx.constraintlayout.widget.ConstraintLayout>
```


2. 뷰페이지 제어하기

- (14-15 line) : 3개의 페이지를 만들기 위한 크기 3의 리스트 생성
- (17 line) : Adapter 할당
- (18 line) : 스와이프 방향 설정(Horizontal – 좌,우)

MainActivity.kt

```
9 class MainActivity : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         val binding = ActivityMainBinding.inflate(layoutInflater)
13         setContentView(binding.root)
14         val datas = mutableListOf<String>()
15         for(i in 1..3){datas.add("Item $i")}
16
17         뷰페이지에 어댑터를 연결, 방향 설정
18     }
19 }
20 }
```

3. 리사이클러뷰 어댑터 활용하기

- (19-22 line) : 페이지 별 배경색 지정

```
9  class MyPagerViewHolder(val binding: ItemPagerBinding) : RecyclerView.ViewHolder(binding.root)
10
11  class MyPagerAdapter(val datas: MutableList<String>) : RecyclerView.Adapter<RecyclerView.ViewHolder>(){
12      override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder =
13          MyPagerViewHolder(ItemPagerBinding.inflate(LayoutInflater.from(parent.context), parent, attachToParent: false))
14
15      override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
16          val binding = (holder as MyPagerViewHolder).binding
17
18          binding.itemPagerTextView.text = datas[position]
19          when (position % 3) {
20              0 -> binding.
21              1 -> binding.
22              2 -> binding.
23          }
24      }
25
26      override fun getItemCount(): Int {
27          return datas.size
28      }
29  }
```

백그라운드 색상 변경

예제 4 – drawerLayout

❖ 토글 레이아웃 구현

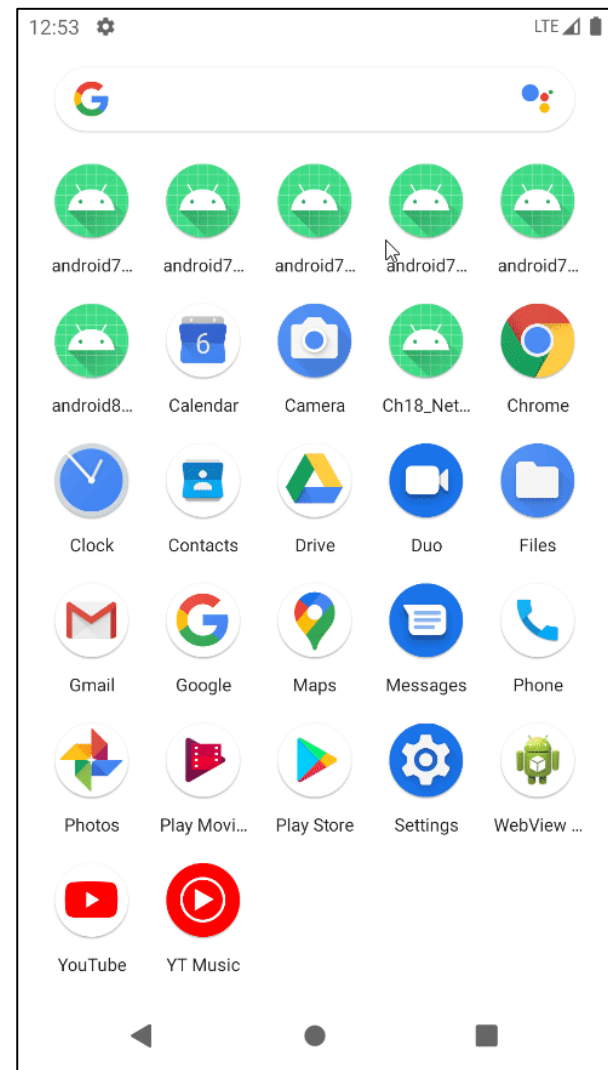
1. 레이아웃 구성하기
2. 기타 앱 리소스 구현하기
3. 드로어 완성하기

Build.gradle(:app)

```
6  android {
7      compileSdk 32
8
9      defaultConfig {...}
10
11      buildFeatures{
12          viewBinding = true
13          dataBinding = true
14      }
15
16      buildTypes {...}
17      compileOptions {...}
18      kotlinOptions {jvmTarget = '1.8'}
19  }
20
21  dependencies {
22      implementation 'androidx.recyclerview:recyclerview:1.1.0'
23      implementation 'androidx.core:core-ktx:1.7.0'
24      implementation 'androidx.appcompat:appcompat:1.4.1'
```

영역 삽입

내용 추가



예제 4 – drawerLayout

1. 레이아웃 구성하기

- (7-17 line) : 액티비티 기본 화면(메인)
- (18-24 line) : 드로어레이아웃 화면

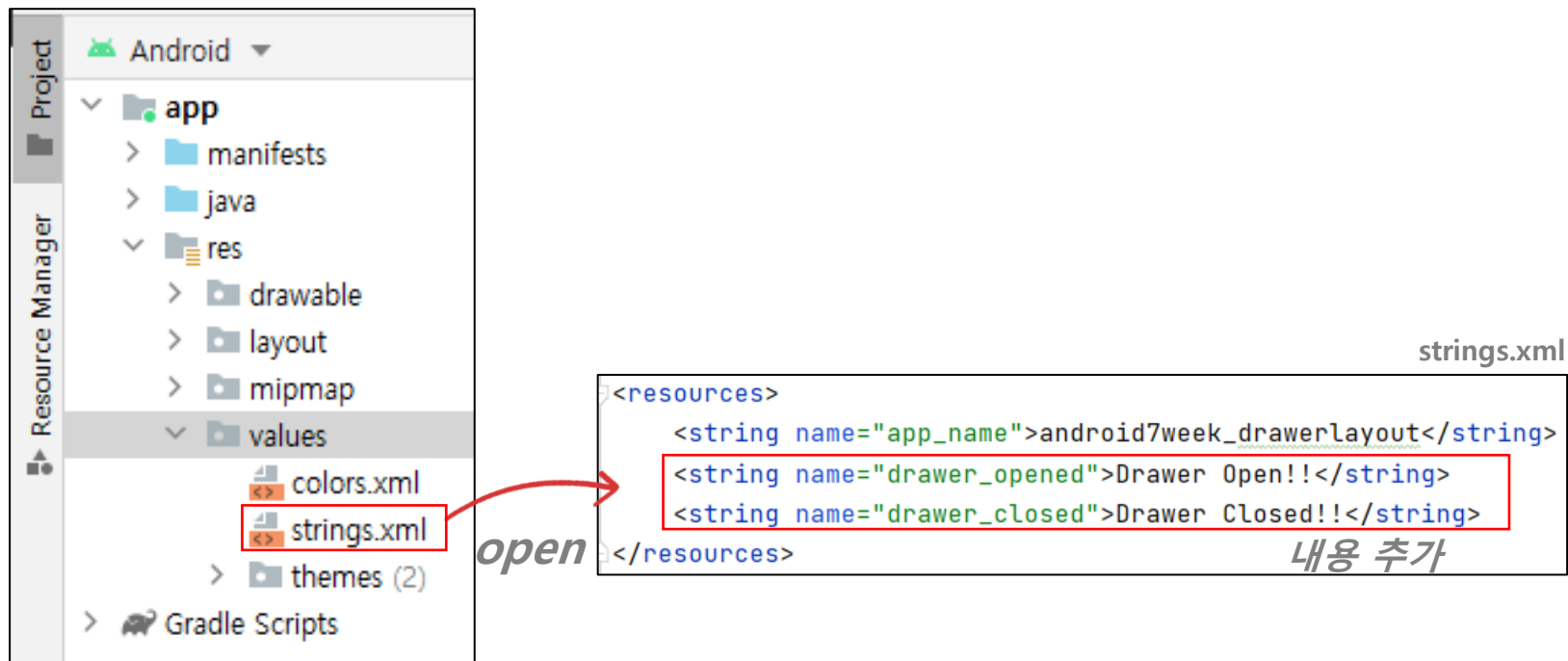
activity_main.xml

```
2 <androidx.drawerlayout.widget.DrawerLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:id="@+id/drawer">
7     <LinearLayout
8         android:layout_width="match_parent"
9         android:layout_height="match_parent"
10        android:id="@+id/linear"
11        android:orientation="horizontal"
12        android:gravity="center">
13        <TextView
14            android:layout_width="wrap_content"
15            android:layout_height="wrap_content"
16            android:text="Main Activity!!"/>
17    </LinearLayout>
18    <TextView
19        android:layout_gravity="start"
20        android:gravity="center"
21        android:layout_height="match_parent"
22        android:layout_width="400px"
23        android:background="@color/purple_200"
24        android:text="I am Drawer!!"/>
25 </androidx.drawerlayout.widget.DrawerLayout>
```

예제 4 – drawerLayout

2. 기타 앱 리소스 구현하기

- 앱에서 자주 사용/출력되는 문자열을 저장하는 파일



3. 드로어 완성하기

- (16-17 line) : 드로어 메뉴 토글 버튼을 선언한다.
- (18 line) : 기존에는 기본(←) 버튼 아이콘이 상단 좌측에 출력된다.
- (19 line) : 상단 좌측의 (←) 버튼을 내비게이션 아이콘(≡)으로 변경한다.
- (22-24 line) : 토글 버튼을 클릭 이벤트 발생 시 드로어가 출력되게 한다.

```
9 class MainActivity : AppCompatActivity() {
10     lateinit var toggle: ActionBarDrawerToggle
11     override fun onCreate(savedInstanceState: Bundle?) {
12         var binding = ActivityMainBinding.inflate(layoutInflater)
13         super.onCreate(savedInstanceState)
14         setContentView(binding.root)
15
16         toggle = ActionBarDrawerToggle(activity: this, binding.drawer,
17             R.string.drawer_opened, R.string.drawer_closed)
18         supportActionBar?.setDisplayHomeAsUpEnabled(true)
19         toggle.syncState()
20     }
21     override fun onOptionsItemSelected(item: MenuItem): Boolean {
22         
23             Drawer 제어
24         
25     }
26 }
```



ActionBar가 나타나지 않을 경우
AndroidManifest.xml에서 테마를 변경

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="week4_ex4"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar"
    tools:targetApi="31">
    <activity
```

❖ 예제 1~4 개발 결과물 통합 구현

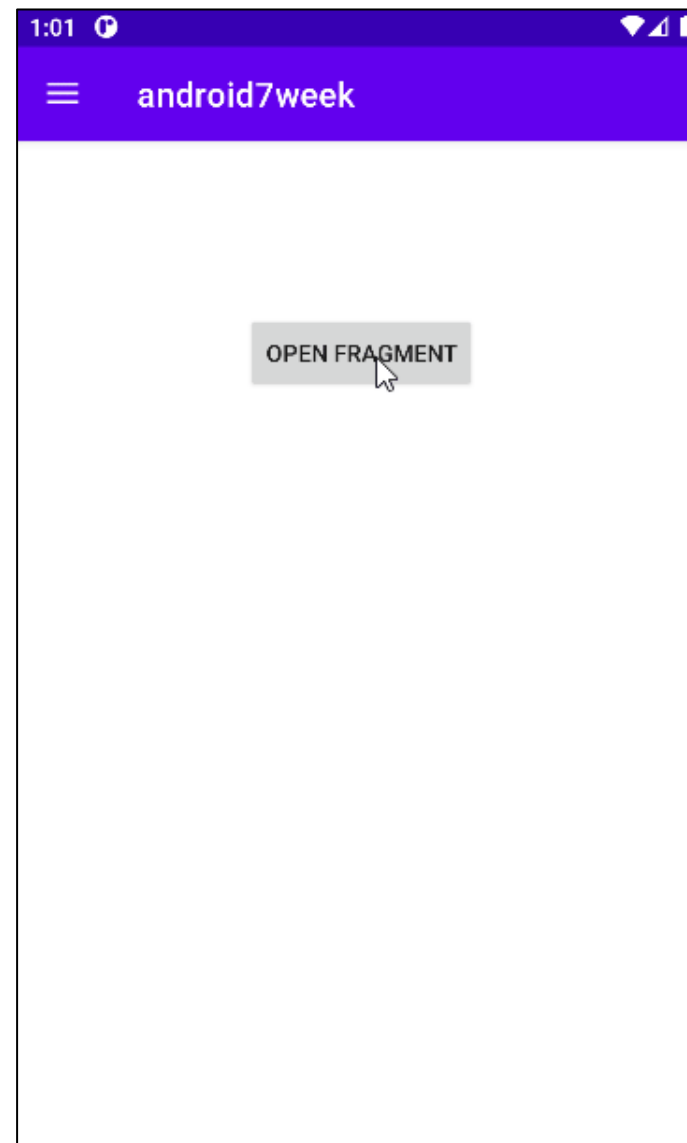
- <기초(따라하기)>에서 구현했던 모든 모듈을 통합하여 하나의 어플리케이션으로 제작
- 조건 : 양방향 액티비티, 인텐트를 활용하지 않고 구현
 1. MainActivity : Fragment [On/Off Switch] 구현
 2. SubActivity : RecyclerView [Item List] 구현
 3. DrawerLayout 동작 구현
 4. ViewPager2를 통해 MainActivity, SubActivity를 스와이프 이동 구현

Build.gradle(app)

```
6  android {
7      compileSdk 32
8
9      defaultConfig {...}
18
19      buildFeatures{
20          viewBinding = true
21          dataBinding = true
22      }
23
24      buildTypes {...}
25      compileOptions {...}
26      kotlinOptions {jvmTarget = '1.8'}
27  }
28
29  dependencies {
30      implementation 'androidx.recyclerview:recyclerview:1.1.0'
31      implementation 'androidx.core:core-ktx:1.7.0'
32      implementation 'androidx.appcompat:appcompat:1.4.1'
```

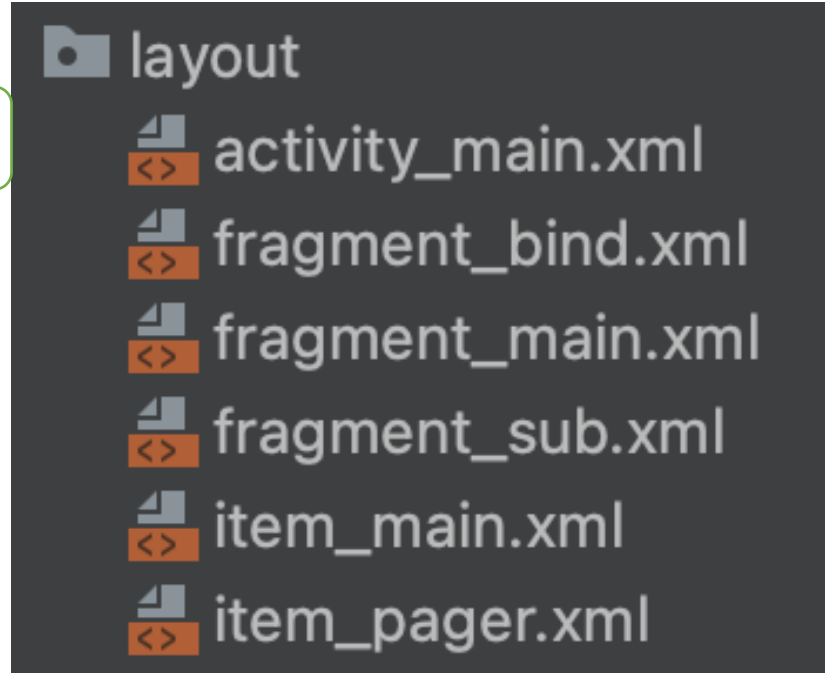
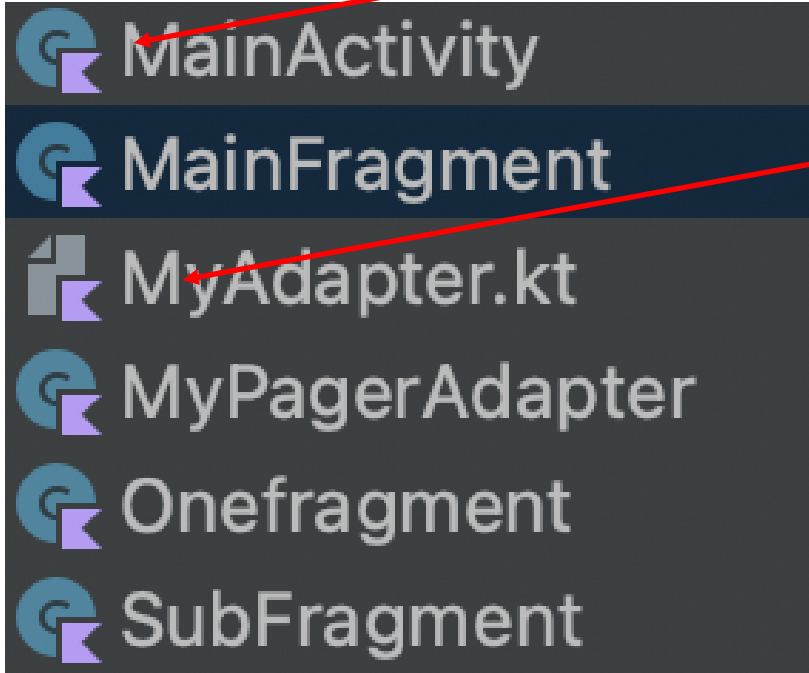
영역 삽입

내용 추가



Class

Kotlin File



예제 5 – MainActivity

```
class MainActivity : AppCompatActivity() {  
    lateinit var toggle : ActionBarDrawerToggle  
    override fun onCreate(savedInstanceState: Bundle?) {  
        var binding = ActivityMainBinding.inflate(layoutInflater)  
        super.onCreate(savedInstanceState)  
        setContentView(binding.root)  
    }  
}
```

페이지 어댑터 추가, 예제 3번 활용 (25p)

```
val drawer = findViewById<DrawerLayout>(R.id.drawer)
```

Drawer를 위한 액션 바 추가, 예제 4번 활용 (30p)

Drawer 추가

```
    override fun onOptionsItemSelected(item: MenuItem): Boolean {  
        if (toggle.onOptionsItemSelected(item)){return true}  
        return super.onOptionsItemSelected(item)  
    }  
}
```

예제 5 – MyPagerAdapter (ViewPager)

```
class MyPagerAdapter(fragmentActivity: FragmentActivity) : FragmentStateAdapter(fragmentActivity) {  
  
    val fragments : List<Fragment> = listOf(MainFragment(), SubFragment())  
    override fun getItemCount(): Int = 1000  
  
    override fun createFragment(position: Int): Fragment {  
  
        if (position % 2 == 0) {  
            return MainFragment()  
        }  
        else {  
            return SubFragment()  
        }  
    }  
}
```

옆으로 넘길 viewPager 횟수

1번 viewPager

2번 viewPager

예제 5 – MainFragment (1st ViewPage)

```
class MainFragment : Fragment() {  
    override fun onCreateView(  
        inflater: LayoutInflater, container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        // Inflate the layout for this fragment  
        var view = inflater.inflate(R.layout.fragment_main, container, attachToRoot: false)  
        val fragmentManager = requireActivity().supportFragmentManager  
        var onClick = false  
        fragButton.setOnClickListener { it: View!  
            if (onClick) {  
                onClick = false  
                val transaction = fragmentManager.beginTransaction()  
                  
            }else {  
                onClick = true  
                  
            }  
        }  
        return view  
    }  
}
```

Id를 이용해서 fragButton 가져오기

FragmentManager를 이용하여 fragment를 추가

FragmentManager를 이용하여 fragment를 삭제

예제 5 – SubFragment (RecyclerView , 2nd viewPage)

```
class SubFragment : Fragment() {  
  
    override fun onCreateView(  
        inflater: LayoutInflater, container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        val view = inflater.inflate(R.layout.fragment_sub, container, attachToRoot: false)  
  
        데이터를 생성, 개수는 20개 예제 2번 이용(21p)  
        RecyclerView 가져오기  
  
        recyclerView의 layout manager, adapter, item decoration 설정 (21p)  
  
        return view  
    }  
}
```

예제 5 – MyAdapter

```
class MyViewHolder(val binding: ItemMainBinding): RecyclerView.ViewHolder(binding.root)

class MyAdapter(val datas: MutableList<String>) : RecyclerView.Adapter<RecyclerView.ViewHolder>() {
    override fun getItemCount(): Int = datas.size
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder = MyViewHolder(
        ItemMainBinding.inflate(LayoutInflater.from(parent.context), parent, attachToParent: false))

    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int){
        Log.d( tag: "kkang", msg: "onBindViewHolder : $position")
        val binding = (holder as MyViewHolder).binding
    }
}
```

항목 뷰를 가지는 역할

항목 구성자

항목 뷰를 가지는 뷰 홀더를
준비하기 위해 자동 호출

데이터를 뷰에 표시 예제2번 응용 (22p)

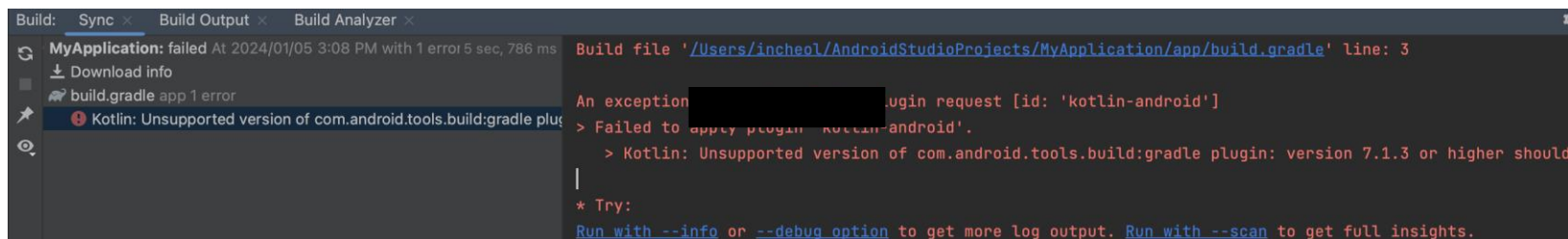
예제 5 – Onefragment

```
class Onefragment: Fragment() {  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        return inflater.inflate(R.layout.fragment_bind, container, attachToRoot: false)  
    }  
}
```

가상 데이터 준비

버전 에러 해결 방법

Gradle plugin 에러 (build gradle project)



Build: Sync x Build Output x Build Analyzer x

MyApplication: failed At 2024/01/05 3:08 PM with 1 error 5 sec, 786 ms

Download info

build.gradle app 1 error

Kotlin: Unsupported version of com.android.tools.build:gradle plugin request [id: 'kotlin-android']

> Failed to apply plugin 'kotlin-android'.

> Kotlin: Unsupported version of com.android.tools.build:gradle plugin: version 7.1.3 or higher should be used.

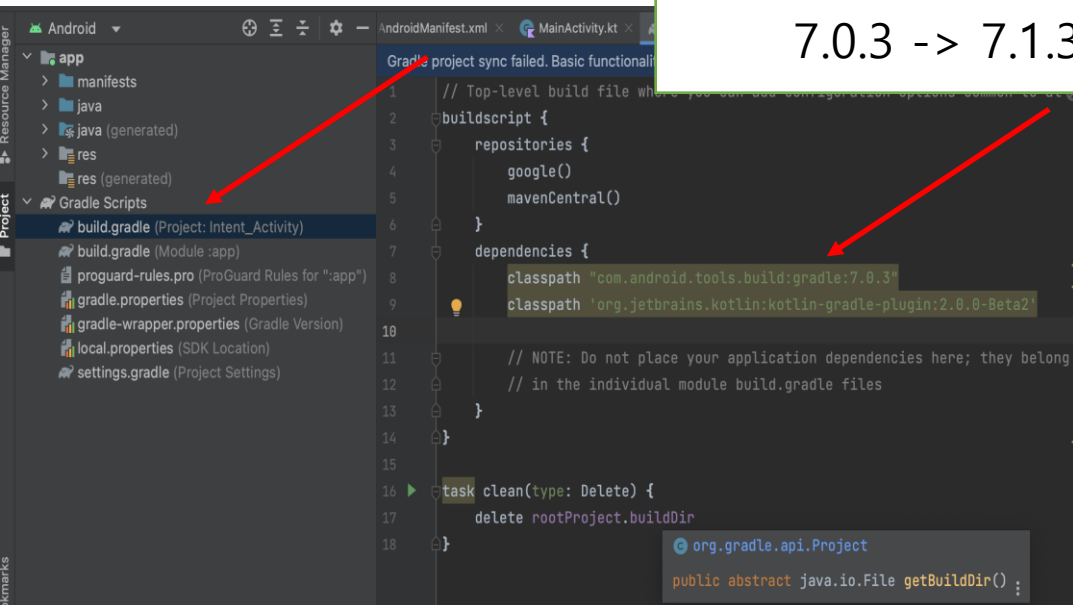
* Try:

Run with --info or --debug option to get more log output. Run with --scan to get full insights.

7.1.3 or higher

7.0.3 -> 7.1.3

버전 수정 후
sync하기



AndroidManifest.xml x MainActivity.kt x

app

manifests

java

java (generated)

res

res (generated)

Gradle Scripts

build.gradle (Project: Intent_Activity)

build.gradle (Module :app)

proguard-rules.pro (ProGuard Rules for ":app")

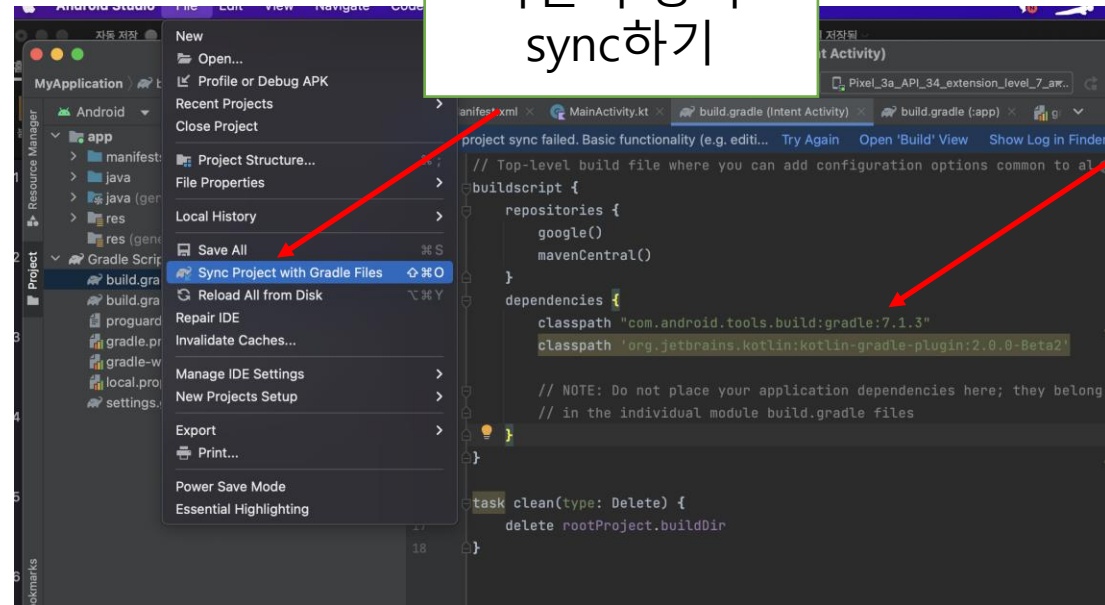
gradle.properties (Project Properties)

gradle-wrapper.properties (Gradle Version)

local.properties (SDK Location)

settings.gradle (Project Settings)

```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2
3 buildscript {
4     repositories {
5         google()
6         mavenCentral()
7     }
8     dependencies {
9         classpath "com.android.tools.build:gradle:7.0.3"
10        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:2.0.0-Beta2"
11    }
12
13    // NOTE: Do not place your application dependencies here; they belong
14    // in the individual module build.gradle files
15
16 task clean(type: Delete) {
17     delete rootProject.buildDir
18 }
```



AndroidManifest.xml x MainActivity.kt x

app

manifests

java

java (generated)

res

res (generated)

Gradle Scripts

build.gradle (Project: Intent_Activity)

build.gradle (Module :app)

proguard-rules.pro (ProGuard Rules for ":app")

gradle.properties (Project Properties)

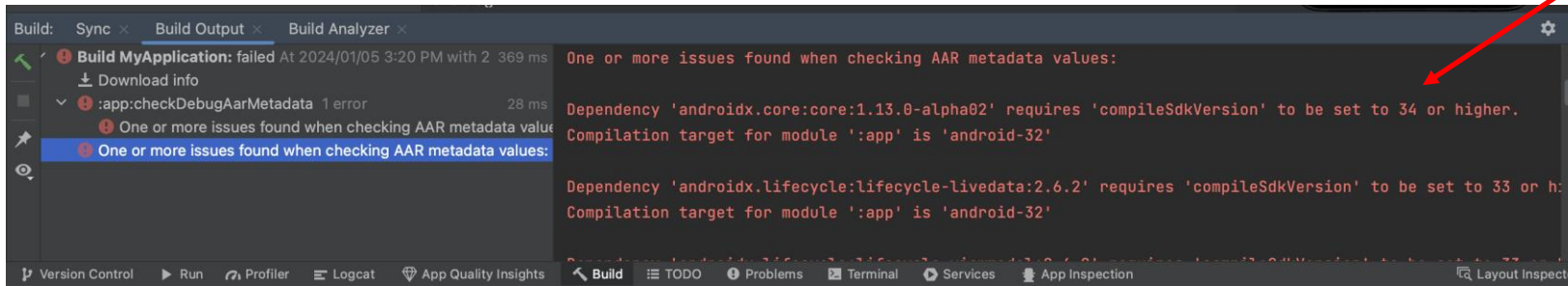
gradle-wrapper.properties (Gradle Version)

local.properties (SDK Location)

settings.gradle (Project Settings)

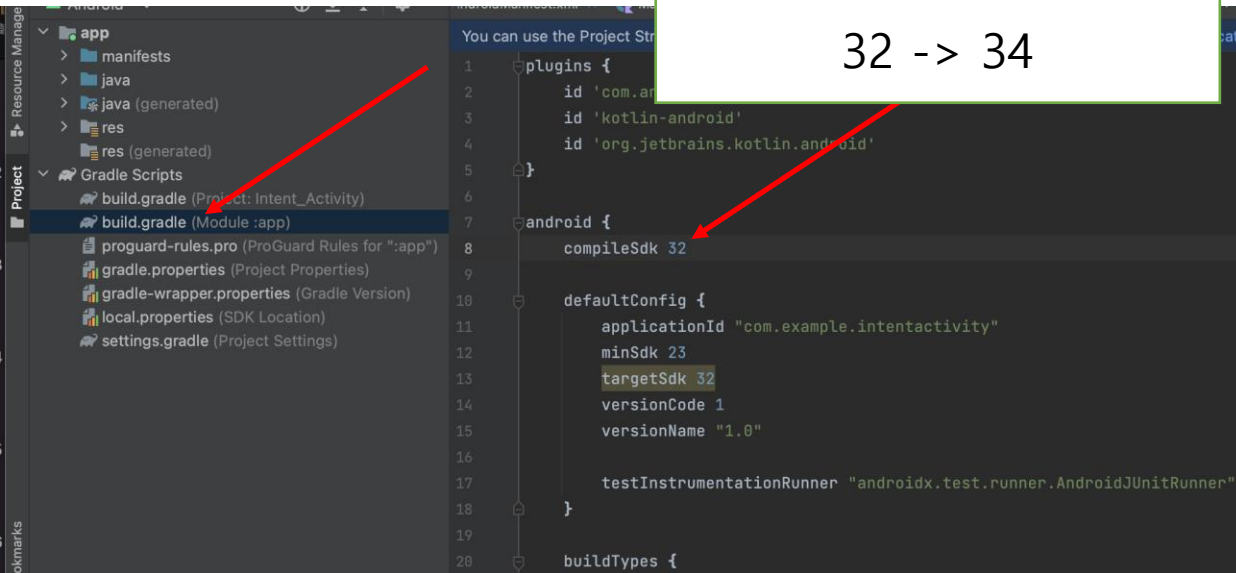
```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules.
2
3 buildscript {
4     repositories {
5         google()
6         mavenCentral()
7     }
8     dependencies {
9         classpath "com.android.tools.build:gradle:7.1.3"
10        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:2.0.0-Beta2"
11    }
12
13    // NOTE: Do not place your application dependencies here; they belong
14    // in the individual module build.gradle files
15
16 task clean(type: Delete) {
17     delete rootProject.buildDir
18 }
```


SDK 버전 에러 (build gradle Module:app)

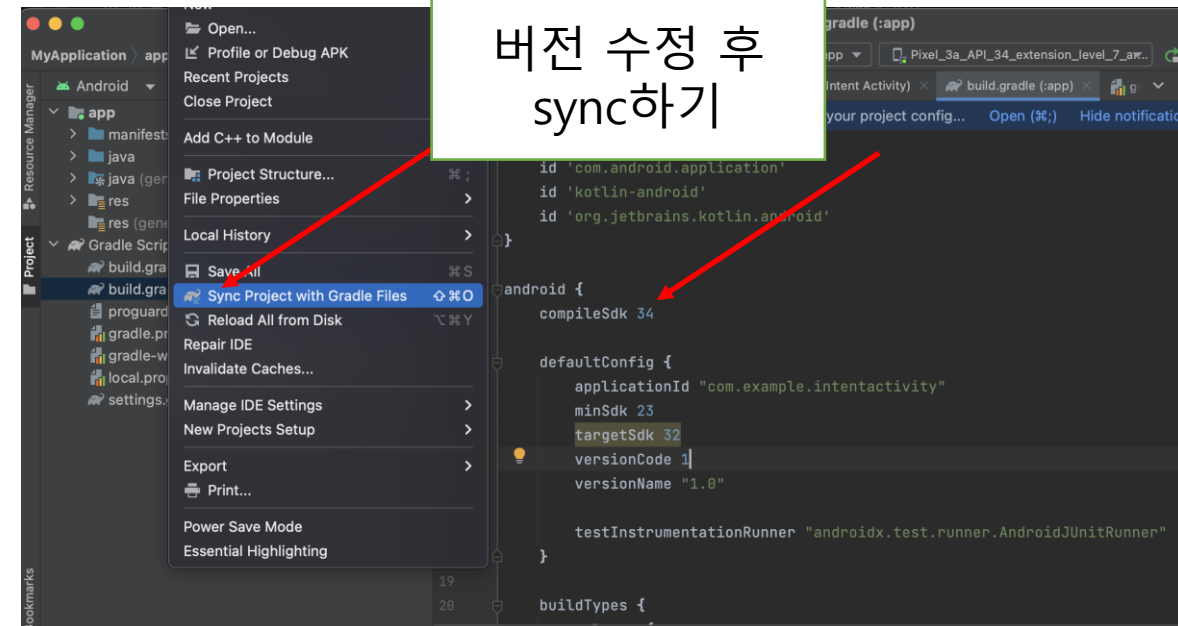


34 or higher

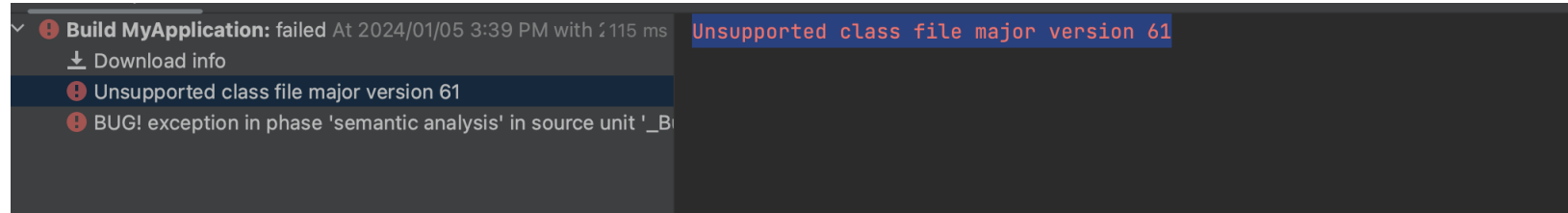
32 -> 34



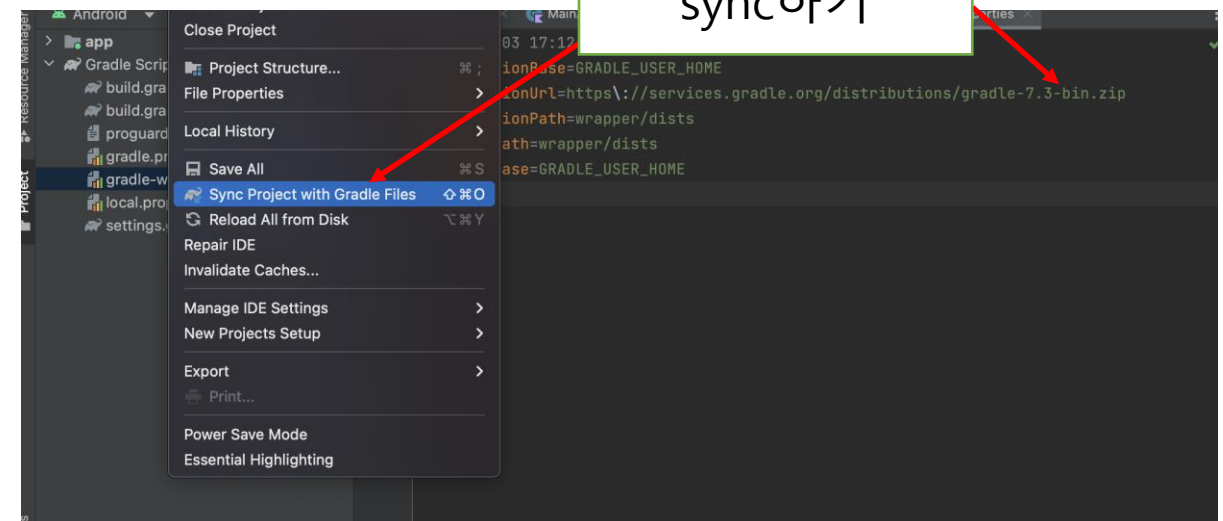
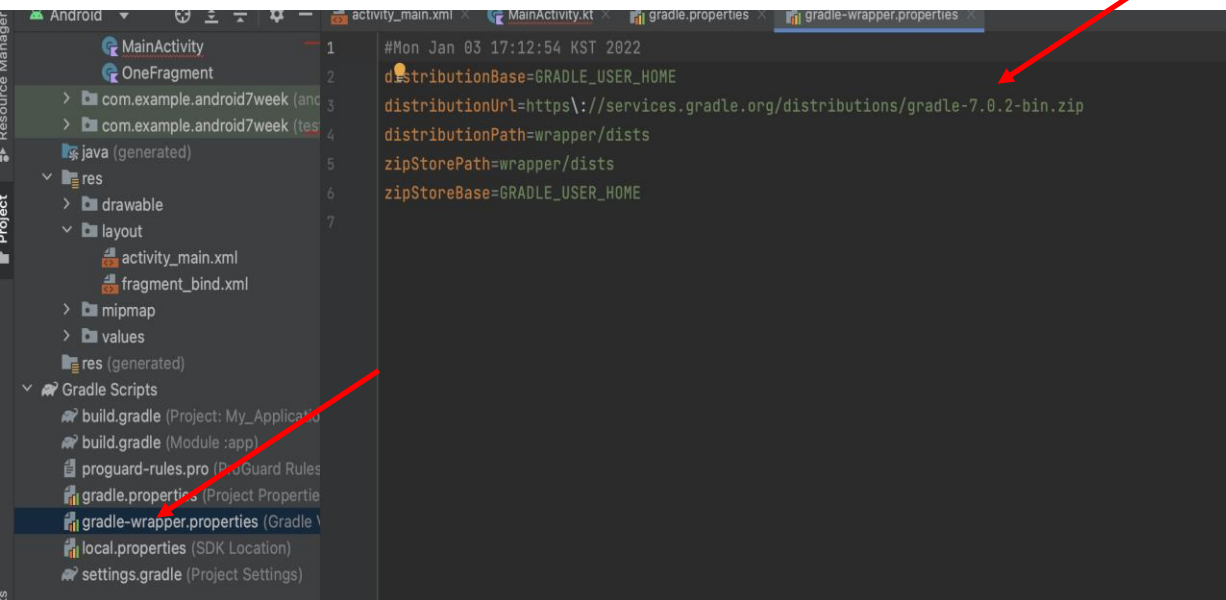
버전 수정 후
sync하기



SDK 버전 에러 (gradle-wrapper.properties)



7.0.2 -> 7.3



버전 수정 후
sync하기