고급 widget과 Fragment : RecyclerView

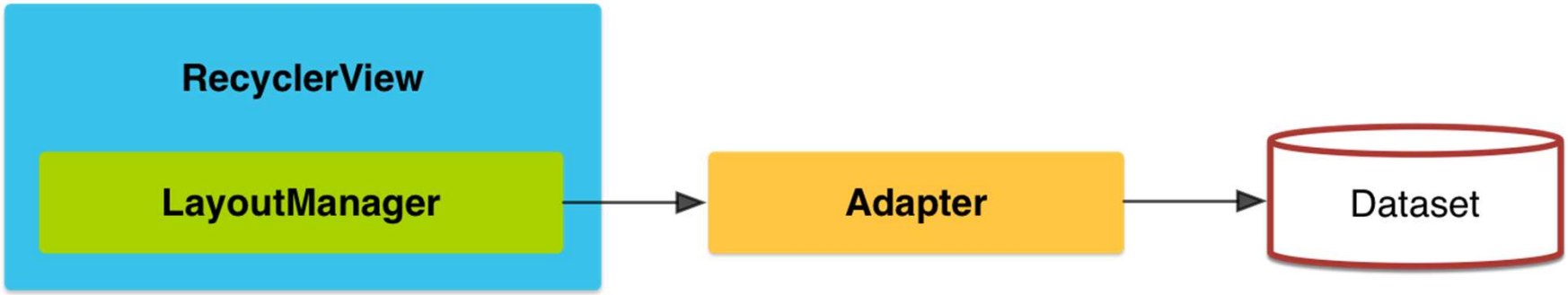
Mobile Software

2019 Fall

RecyclerView 는?

- RecyclerView는 ListView에 비해 보다 유연하고 향상된 성능 제공
  - 5.0(API 21)부터 추가됨.
  - 화면을 스크롤 할 때 기존 view가 스크롤되어 화면을 벗어나도 해당 view를 버리지 않고 재활용(recycle)
- ListView에 비해 달라진 점
  - **LayoutManager :** 각 항목(item)의 view를 RecyclerView에 배치
    - Item에 속한 view의 재사용과 관련된 판단
      - LinearLayoutManager (수평 또는 수직 방향으로 일렬로 배치)
      - GridLayoutManager (바둑판 모양 배치)
      - StaggeredGridLayoutManager (크기가 다른 사각형 배치)
  - **ViewHolder**
    - RecyclerView에 출력되는 Item : ViewHolder 클래스의 인스턴스
    - Item에서 포함된 view와 레이아웃

RecyclerView 동작 모델




```
val recyclerView =
    findViewById<RecyclerView>(R.id.recyclerView)

layoutManager = LinearLayoutManager(this)
recyclerView.layoutManager = layoutManager

adapter = MyAdapter(listItems)
recyclerView.adapter = adapter
```

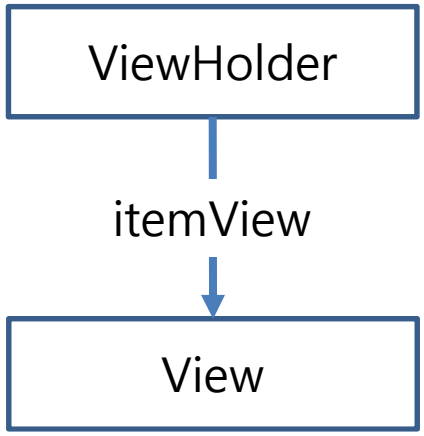
RecyclerView와 Adapter

- RecyclerView는 직접 view객체를 생성하지 않는다.

- RecyclerView는 화면에 보여 줄 객체가 필요할 때 Adapter와 소통한다.
  - Adapter: **onCreateViewHolder**
    - ViewHolder **레이아웃을 생성 ➔ view 객체 참조**
  - Adapter: **onBindViewHolder**
    - 리스트에 속한 항목을 하나씩 참조
      - 이를 ViewHolder의 View에 결합(bind)시킴
    - **ViewHolder가 hold하고 있는 view 에 속성 할당**
  - Adapter: **getItemCount**
    - 리스트 항목 전체 수

ViewHolder

- View를 유지(hold)하는 역할



itemView는 **RecyclerView.ViewHolder** 클래스의 property

```
class MyViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
    var textView: TextView
            = itemView.findViewById(R.id.textView)

}
```

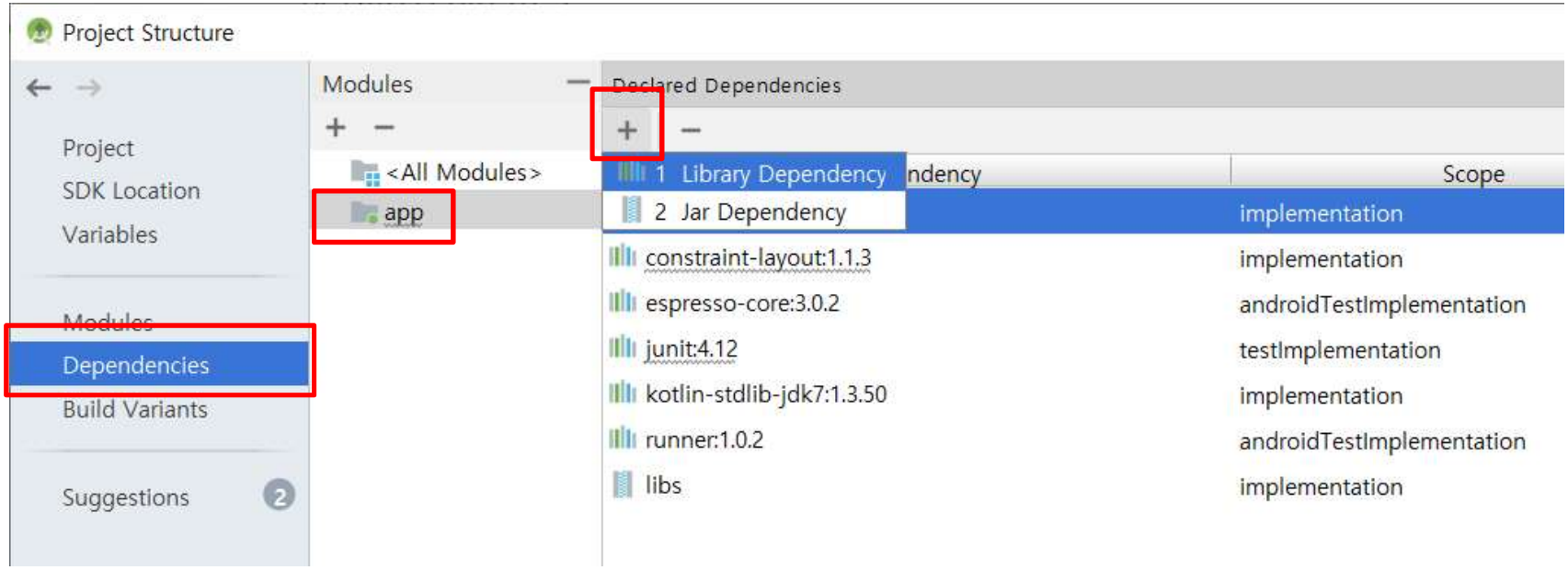
실습 준비

- 새 프로젝트 생성
  - **Application name**
    - **Ch9 project**
  - **Target Android Devices**
    - **Phone and Tablet**
      - minimum SDK **– API 26** 이상
  - **Activity**
    - **Empty Activity**

- 자동 생성된 layout은 **ConstraintLayout**
  - TextView 삭제

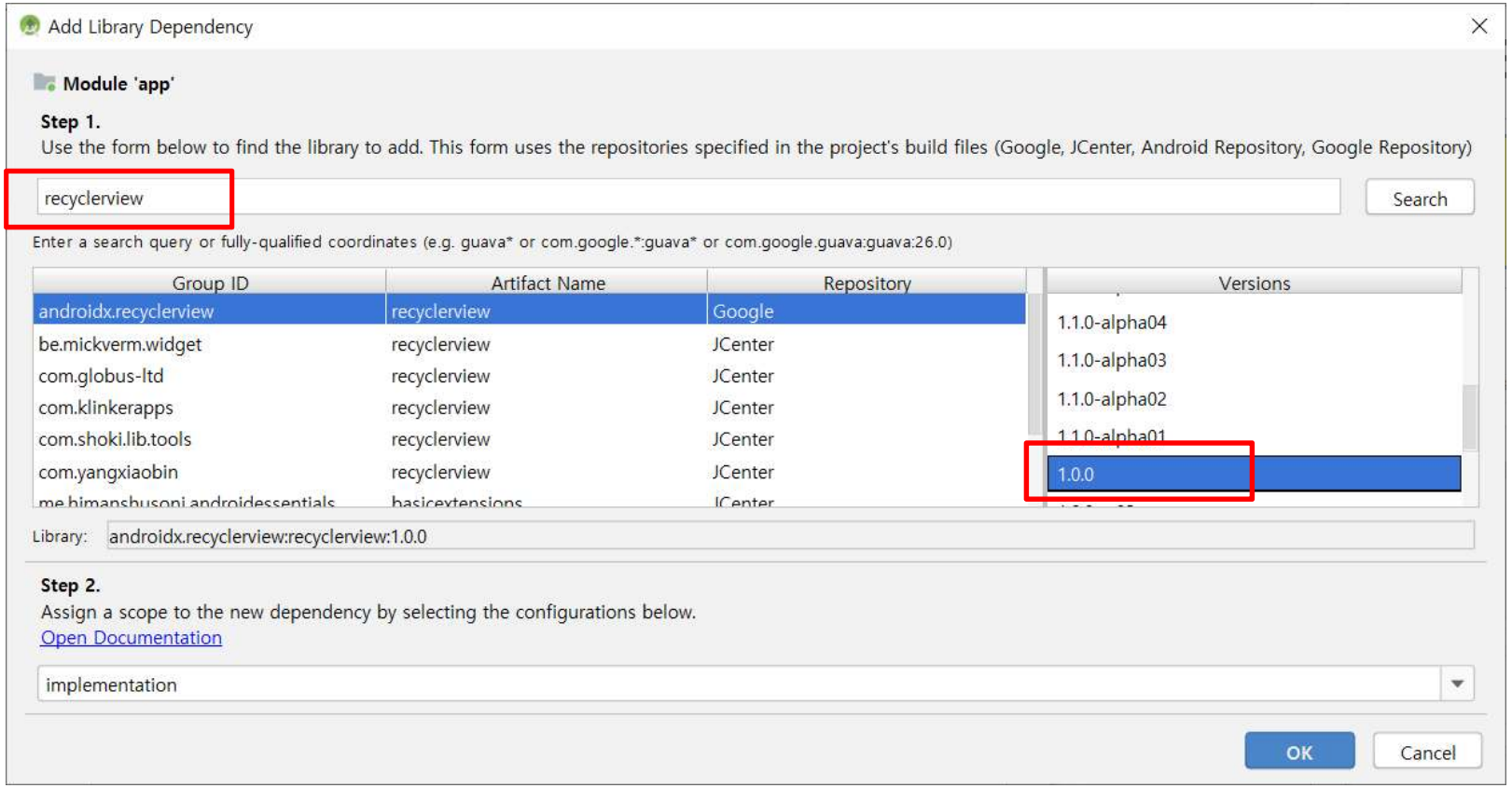
recyclerview-v7 라이브러리 추가(1/2)

File > Project Structure
화면 왼쪽 **Modules : app** 선택
→ 화면 오른쪽 '+' 클릭 → **Library dependency** 선택



recyclerview-v7 라이브러리 추가(2/2)

검색 창에 "**recyclerview**" → Enter
→ **androidx.recyclerview-v7** → **1.0.0** 선택 → OK



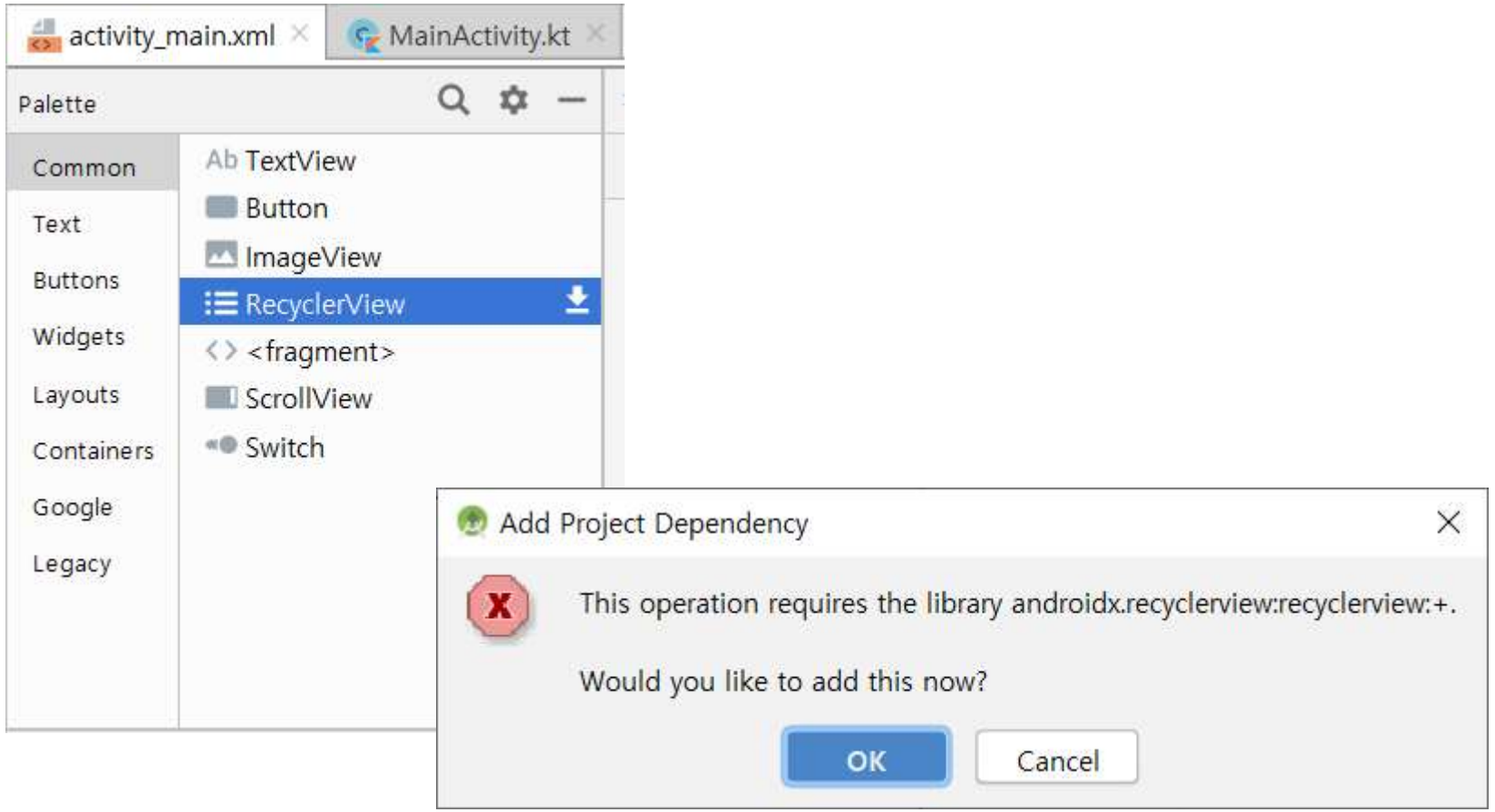
recyclerview-v7 라이브러리 추가: 방법 2

Project 창 > Gradle Scripts > build.gradle (**Module:app**) 클릭

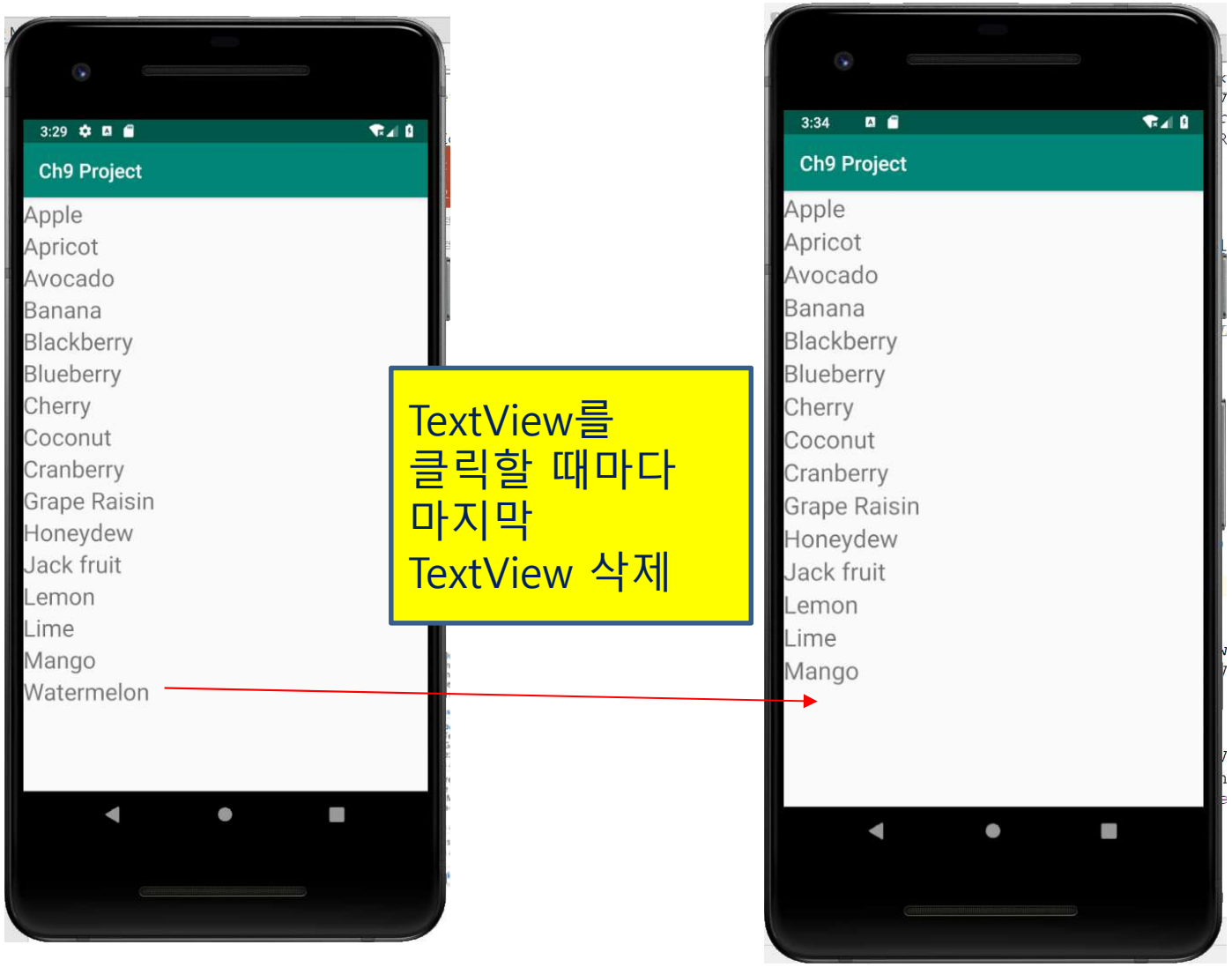
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.core:core-ktx:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'androidx.recyclerview:recyclerview:1.0.0'
}
```

위 방법보다는 build.gradle (**Module:app**) 에서
implementation ... 문장을 직접 입력하고
Sync Now를 누르는 게 더 빠른 방법.
단, 이 경우 정확한 라이브러리 버전을 알고 있어야 함

recyclerview-v7 라이브러리 추가: 방법 3



실습 1: RecyclerView 이해



실습 1: RecyclerView 포함

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">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        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" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

소스코드 – 1쪽

실습 1: Item Layout

item.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="wrap_content"
        android:orientation="vertical">
    <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView"
            android:textSize="24sp"/>
</LinearLayout>
```

match_parent로 설정하면 다른 item이 안 보일 수 있음

View item은 하나
➔ TextView

소스코드 – 1쪽

실습 1: Adapter 정의(1/2)

MyAdapter.kt

```
class MyAdapter(list:ArrayList<String>)
        : RecyclerView.Adapter<MyAdapter.MyViewHolder>(){

    private var nameList:ArrayList<String> = list

    // ViewHolder 객체 생성. 이 때 item 출력을 위한 레이아웃 파일(item.xml)도
    // 팽창(inflate)시킴 -> 화면에 보이도록 함.
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int)
            : MyViewHolder {
        val inflater = LayoutInflater.from(parent.context)
        val view = inflater.inflate(R.layout.item, parent, false)
        return MyViewHolder(view)
    }

    override fun getItemCount(): Int {
        return nameList.size
    }

    // 리스트(nameList)에 속한 item 하나씩 가져 와 출력
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        val name = nameList[position]
        holder.textView.text = name
    }
```

소스코드 – 1~2쪽

실습 1: Adapter 정의(2/2)

MyAdapter.kt

```
class MyAdapter(list:ArrayList<String>)
        : RecyclerView.Adapter<MyAdapter.MyViewHolder>(){

    private var nameList:ArrayList<String> = list

    // ViewHolder 객체 생성. 이 때 item 출력을 위한 레이아웃 파일(item.xml)도
    // 팽창(inflate)시킴 -> 화면에 보이도록 함.
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int)
            : MyViewHolder {...}

    override fun getItemCount(): Int {...}

    // 리스트(nameList)에 속한 item 하나씩 가져 와 출력
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {...}

    class MyViewHolder(itemView: View) :
        RecyclerView.ViewHolder(itemView) {
        var textView: TextView
                = itemView.findViewById(R.id.textView)

    }
}
```

실습 1: Activity

MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    private var layoutManager:
            RecyclerView.LayoutManager? = null
    private var adapter:
            RecyclerView.Adapter<MyAdapter.MyViewHolder>? = null

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

        val listItems = ArrayList<String>()
        listItems.add("Apple")
        listItems.add("Apricot")

        layoutManager = LinearLayoutManager(this)
        recyclerView.layoutManager = layoutManager

        adapter = MyAdapter(listItems)
        recyclerView.adapter = adapter
    }
}
```

소스코드 – 2~3쪽

실습 1(b): array 리소스 할당

MainActivity.kt

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

    // ArrayList는 가변형(mutable).
    // element 추가(add)나 삭제(remove)가 쉬움.
    var listItems = ArrayList<String>()
    // val listItems: ArrayList<String> =
    //          arrayListOf<String>()

    val list:Array<String> =
        resources.getStringArray(R.array.fruits)
    for (element in list)
        listItems.add(element)

    layoutManager = LinearLayoutManager(this)
    recyclerView.layoutManager = layoutManager
```

배열을 정의한 파일 생성
res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="fruits">
        <item>Apple</item>
        <item>Apricot</item>
        <item>Avocado</item>
        <item>Banana</item>
        <item>Blueberry</item>
        <item>Cherry</item>
        <item>Coconut</item>
        <item>Cranberry</item>
        <item>Grape Raisin</item>
        <item>Honeydew</item>
        <item>Jackfruit</item>
        <item>Lemon</item>
        <item>Lime</item>
        <item>Mango</item>
        <item>Watermelon</item>
    </string-array>
</resources>
```

소스코드 – 4,5쪽

소스코드 – 4쪽

실습 1(c): 클릭 이벤트 추가

방법 1

MyAdapter.kt

```
inner class MyViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
    var textView: TextView

    init {
        textView = itemView.findViewById(R.id.textView)
        textView.setOnClickListener {
            var position = adapterPosition
            remove(position)
        }
    }
}

private fun remove(position: Int) {
    nameList.removeAt(position)
    notifyItemRemoved(position)
}
```

소스코드 – 6쪽

실습 1(c): 클릭 이벤트 추가

방법 2

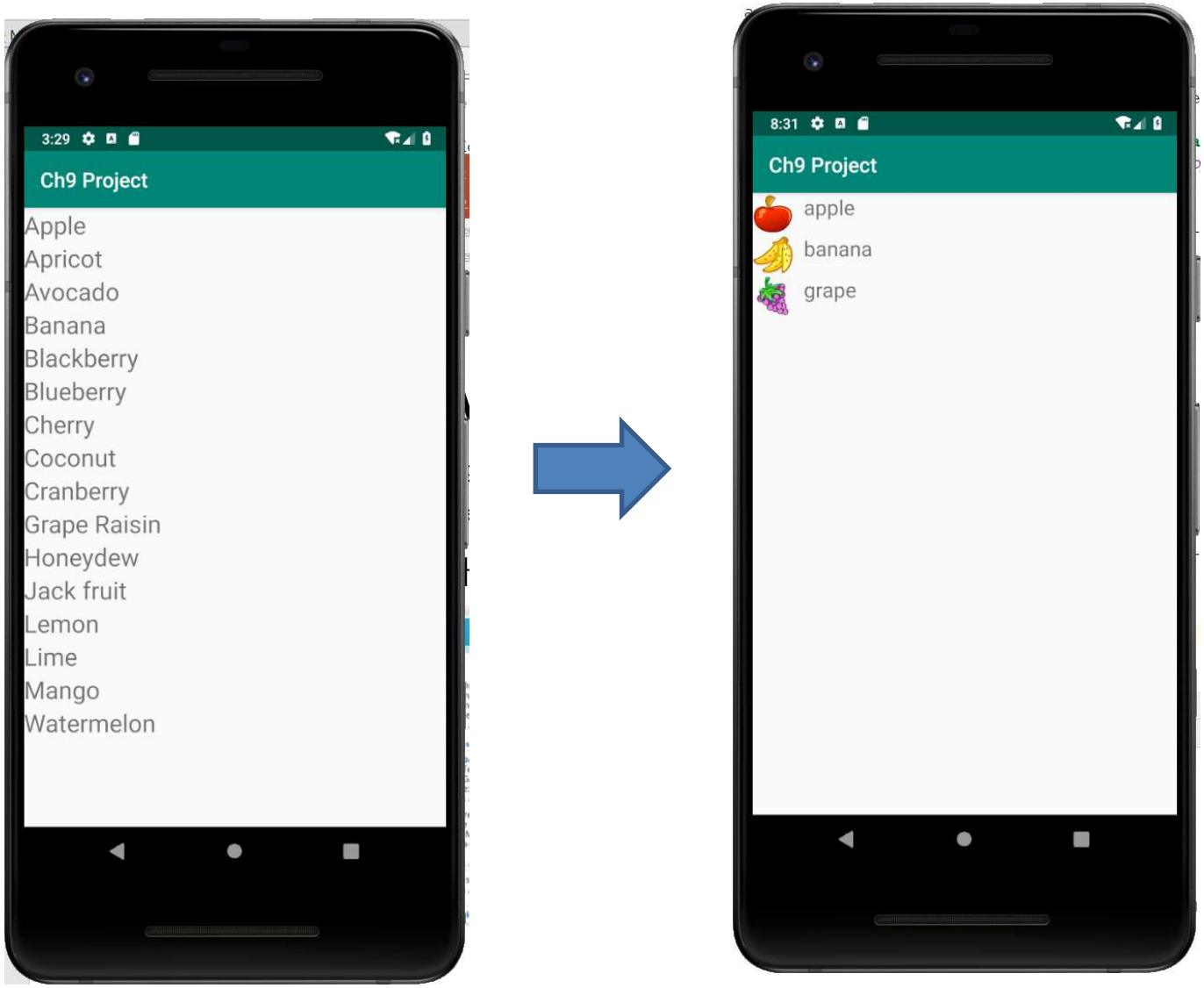
MyAdapter.kt

```
// 리스트(nameList)에 속한 item 하나씩 가져 와 출력
override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
    val name = nameList[position]
    holder.textView.text = name

    holder.itemView.setOnClickListener {
        remove(position)
    }
}

inner class MyViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
    var textView: TextView =
        itemView.findViewById(R.id.textView)
}

private fun remove(position:Int) {
    nameList.removeAt(position)
    notifyItemRemoved(position)
}
```

itemView 대신 textView로 입력하면 exception 발생

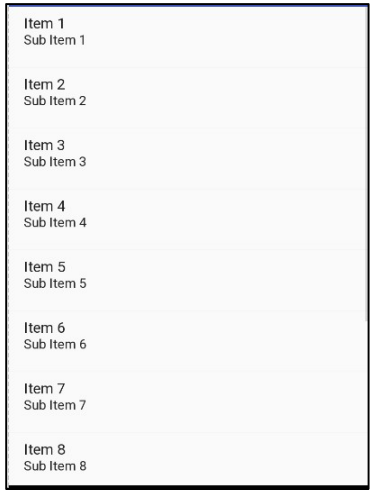
소스코드 – 6쪽

실습 2: Customized RecyclerView (1)



실습 2: 파일 구성

fruit_item.xml — FruitAdapter.kt



activity_main.xml — MainActivity.kt

실습 1과 같음

```
adapter = FruitAdapter()
recyclerView.adapter = adapter
```

실습 2: 한 개 item에 대한 layout

fruit_item.xml

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    <ImageView
            android:id="@+id/imageView"
            android:layout_width="40dp"
            android:layout_height="40dp"
            app:srcCompat="@drawable/apple" />

    <TextView
            android:id="@+id/textView"
            android:layout_width="200dp"
            android:layout_height="40dp"
            android:layout_marginStart="10dp"
            android:gravity="start"
            android:text="TextView"
            android:textSize="20dp" />
</LinearLayout>
```

소스코드 – 7쪽

실습 2: Adapter (1/2)

FruitAdapter.kt

```
class FruitAdapter:
    RecyclerView.Adapter<FruitAdapter.MyViewHolder>() {

    private val titles = arrayOf("apple", "banana", "grape")
    private val images = intArrayOf(R.drawable.apple,
        R.drawable.banana, R.drawable.grape)

    override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int): MyViewHolder {
        val v = LayoutInflater.from(viewGroup.context)
            .inflate(R.layout.fruit_item, viewGroup, false)
        return MyViewHolder(v)
    }

    override fun onBindViewHolder(holder: MyViewHolder, postition: Int) {
        holder.itemText.text = titles[postition]
        holder.itemImage.setImageResource(images[postition])
    }

    override fun getItemCount(): Int {
        return titles.size
    }
```

소스코드 – 7~8쪽

실습 2: Adapter (2/2)

FruitAdapter.kt

```
class FruitAdapter:
    RecyclerView.Adapter<FruitAdapter.MyViewHolder>() {

    private val titles = arrayOf("apple", "banana", "grape")
    private val images = intArrayOf(R.drawable.apple,
        R.drawable.banana, R.drawable.grape)

    override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int)
            : MyViewHolder {...}

    override fun onBindViewHolder(holder: MyViewHolder,
                                  postition: Int) {...}

    override fun getItemCount(): Int {...}

    inner class MyViewHolder(itemView: View)
        : RecyclerView.ViewHolder(itemView) {
        var itemText: TextView
        var itemImage: ImageView

        init {
            itemText = itemView.findViewById(R.id.textView)
            itemImage = itemView.findViewById(R.id.imageView)
        }
    }
}
```

inflate 메소드

inflate

```
View inflate (int resource,
              ViewGroup root,
              boolean attachToRoot)
```

Layout XML파일(*resource*)을
view 객체로 만듦

Parameters	
`resource`	`int`: ID for an XML layout resource to load (e.g., `R.layout.main_page`)
`root`	`ViewGroup`: Optional view to be the parent of the generated hierarchy (if *attachToRoot* is true), or else simply an object that provides a set of LayoutParams values for root of the returned hierarchy (if *attachToRoot* is false.)
`attachToRoot`	`boolean`: Whether the inflated hierarchy should be attached to the root parameter? If false, root is only used to create the correct subclass of LayoutParams for the root view in the XML.

inflate (... , ***null, false***);
➔ root 에 갖다 붙이지(*attach*) 않고 단순히 view만 보여 줌.
inflate (... , ***root, true***);
➔ root로 구성된 hierarchy에 이 view를 추가 (*attachToRoot*)

실습 2: Activity

MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    private var layoutManager:
            RecyclerView.LayoutManager? = null
    private var adapter:
            RecyclerView.Adapter<FruitAdapter.MyViewHolder>? = null

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

        layoutManager = LinearLayoutManager(this)
        recyclerView.layoutManager = layoutManager

        adapter = FruitAdapter()
        recyclerView.adapter = adapter
    }
}
```

소스코드 – 8쪽

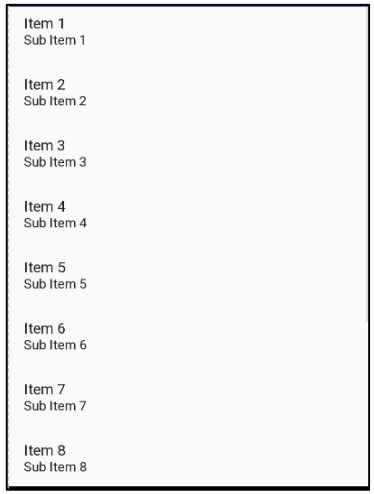
실습 2(b): 데이터 클래스 사용

```
class FruitItem (var name: String,
                 var resId: Int)
```

FruitItem.kt

실습 2와 같음

fruit_item.xml — FruitAdapter.kt



activity_main.xml — MainActivity.kt 실습 2와 같음

실습 1과 같음

```
adapter = FruitAdapter()
recyclerView.adapter = adapter
```

실습 2(b): item 저장을 위한 데이터 클래스

FruitItem.kt

```
class FruitItem(var name: String, var resId: Int)
```

```
FruitItem("apple", R.drawable.apple)
```

소스코드 – 9쪽

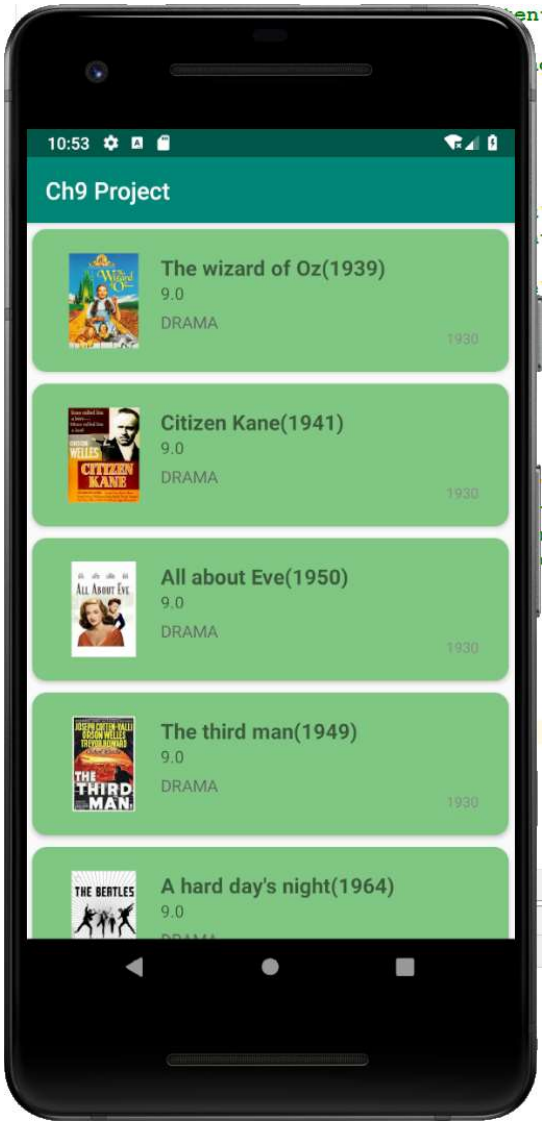
실습 2(b): Adapter

```
class FruitAdapter:
    RecyclerView.Adapter<FruitAdapter.MyViewHolder>() {

    var fruitList = ArrayList<FruitItem>()

    init {
        fruitList.add(FruitItem("apple", R.drawable.apple))
        fruitList.add(FruitItem("banana", R.drawable.banana))
    }

    override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int)
            : MyViewHolder {...}

    override fun onBindViewHolder(holder: MyViewHolder,
                                  postition: Int) {
        var items = fruitList[postition]
        holder.itemText.text = items.name
        holder.itemImage.setImageResource(items.resId)
    }

    override fun getItemCount(): Int {
        return fruitList.size
    }

    inner class MyViewHolder(itemView: View)
        : RecyclerView.ViewHolder(itemView) {...}
}
```

FruitAdapter.kt

소스코드 – 9~10쪽

실습 3: Customized RecyclerView (2)



cardview-v7 라이브러리 추가

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.core:core-ktx:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'androidx.recyclerview:recyclerview:1.0.0'
    implementation 'androidx.cardview:cardview:1.0.0'

}
```

build.gradle (**Module:app**) 에서
implementation ... 문장을 직접 입력하고
Sync Now를 클릭해서 동기화.

실습 3: 파일 구성

```
class MovieItem(var title: String,
                var rating: String,
                var genre: String,
                var year: String,
                var resId: Int)
```

MovieItem.kt

movie_item.xml

MovieAdapter.kt

activity_main.xml

MainActivity.kt

실습 1과 같음

```
adapter = FruitAdapter()
recyclerView.adapter = adapter
```

실습 3: Item Layout (1/2)

movie_item.xml

```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    app:cardBackgroundColor="#81C784"
    app:cardCornerRadius="12dp"
    app:cardElevation="3dp"
    app:contentPadding="4dp"
    android:foreground="?selectableItemBackground"
    android:clickable="true">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">
        <ImageView...>

        <TextView...>

        <TextView...>

        <TextView...>

        <TextView...>
    </RelativeLayout>
</androidx.cardview.widget.CardView>
```

height를 반드시 wrap_content로 바꿔야 함.
그렇지 않으면
한 개의 item 밖에
출력되지 않음.

소스코드 – 11~12쪽

실습 3: Item Layout (2/2)

movie_item.xml

```
<RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp">
    <ImageView
            android:layout_width="80dp"
            android:layout_height="80dp"
            app:srcCompat="@drawable/movie1"
            android:layout_alignParentTop="true"
            android:layout_alignParentStart="true"
            android:id="@+id/image"
            android:layout_marginRight="8dp"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_toRightOf="@+id/image"
            android:id="@+id/title"
            android:text="The wizard of Oz(1939)"
            android:textSize="18sp"
            android:textStyle="bold"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/title"
            android:layout_toRightOf="@+id/image"
            android:id="@+id/rating"
            android:text="9.0"
            android:textSize="14sp"/>
```

The wizard of Oz(1939)

9.0

DRAMA

1939

소스코드 – 11~12쪽

실습 3: item 저장을 위한 데이터 클래스

MovieItem.kt

```
class MovieItem(var title: String,
                var rating: String,
                var genre: String,
                var year: String,
                var resId: Int)
```

```
movieList.add(MovieItem("The wizard of Oz(1939)",
    "9.0", "DRAMA", "1930", R.drawable.movie1))
```

소스코드 – 12쪽

실습 3: Adapter 정의 (1/2)

```
class MovieAdapter :
    RecyclerView.Adapter<MovieAdapter.MyViewHolder>() {

    var movieList = ArrayList<MovieItem>()

    init {
        movieList.add(MovieItem("The wizard of Oz(1939)",
            "9.0", "DRAMA", "1930", R.drawable.movie1))
        movieList.add(MovieItem("Citizen Kane(1941)",
            "9.0", "DRAMA", "1930", R.drawable.movie2))
    }

    override fun onCreateViewHolder(viewGroup: ViewGroup, i: Int)
            : MyViewHolder {
        val v = LayoutInflater.from(viewGroup.context)
            .inflate(R.layout.movie_item, viewGroup, false)
        return MyViewHolder(v)
    }
```

소스코드 – 12~14쪽

실습 3: Adapter 정의 (2/2)

```
override fun onBindViewHolder(holder: MyViewHolder,
                              postition: Int) {
    var items = movieList[postition]
    holder.itemTitle.text = items.title
    holder.itemRating.text = items.rating
    holder.itemGenre.text = items.genre
    holder.itemYear.text = items.year
    holder.itemImage.setImageResource(items.resId)
}

override fun getItemCount(): Int {
    return movieList.size
}

inner class MyViewHolder(itemView: View)
    : RecyclerView.ViewHolder(itemView) {
    var itemImage: ImageView
    var itemTitle: TextView
    var itemRating: TextView
    var itemGenre: TextView
    var itemYear: TextView

    init {
        itemImage = itemView.findViewById(R.id.image)
        itemTitle = itemView.findViewById(R.id.title)
        itemRating = itemView.findViewById(R.id.rating)
        itemGenre = itemView.findViewById(R.id.genre)
        itemYear = itemView.findViewById(R.id.releaseYear)
    }
}
```

실습 3: Activity

MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    private var layoutManager:
            RecyclerView.LayoutManager? = null
    private var adapter:
            RecyclerView.Adapter<MovieAdapter.MyViewHolder>? = null

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

        layoutManager = LinearLayoutManager(this)
        recyclerView.layoutManager = layoutManager

        adapter = MovieAdapter()
        recyclerView.adapter = adapter
    }
}
```

소스코드 – 14쪽