AdapterView의 일종

RecyclerView 01

- http://developer.android.com

목차

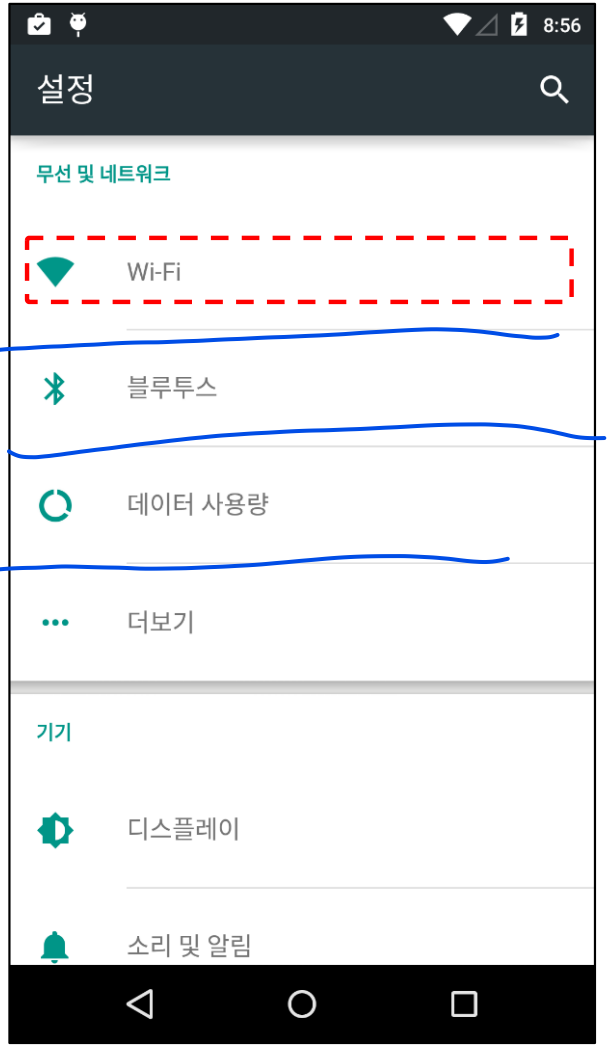
- AdapterView의 개요
  - RecycleView의 상위 클래스
- Adapter 클래스
  - 같이 써야 함
  - + data
- AdapterView의 사용 개요
  - 목록형, 갤러리형
- RecyclerView 의 사용 절차
- 이벤트 처리
- 항목 편집 처리

View Group의 일종

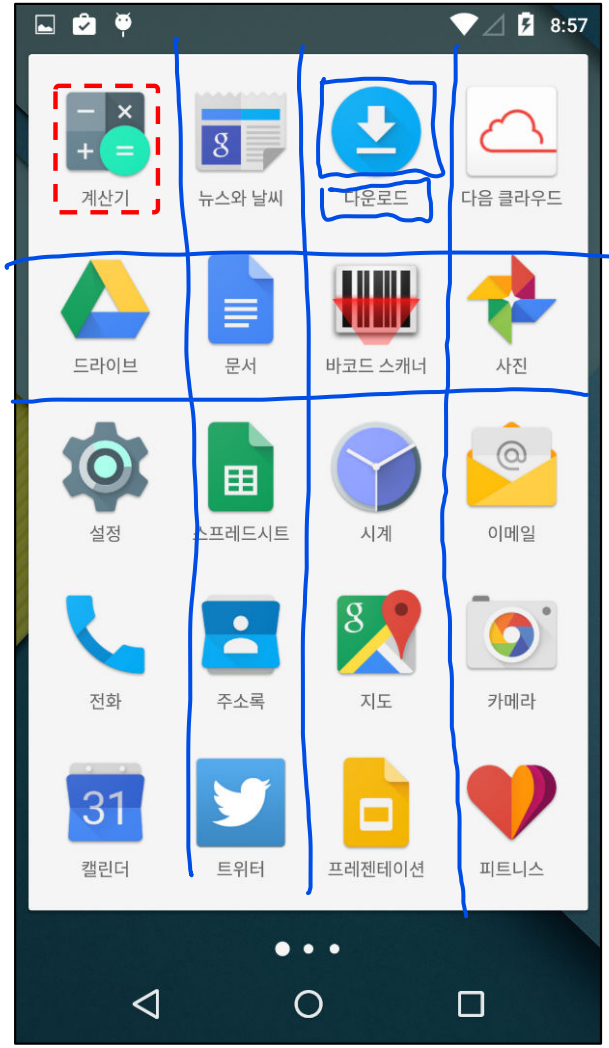
- View : 화면에 보여주는 것
  - 위젯 : 화면의 button, editText 등
- View Group : 일반적인 View에서 다른 View를 담는 것 (e.g. Layout)

AdapterView 개요 1 일정한 공간에!

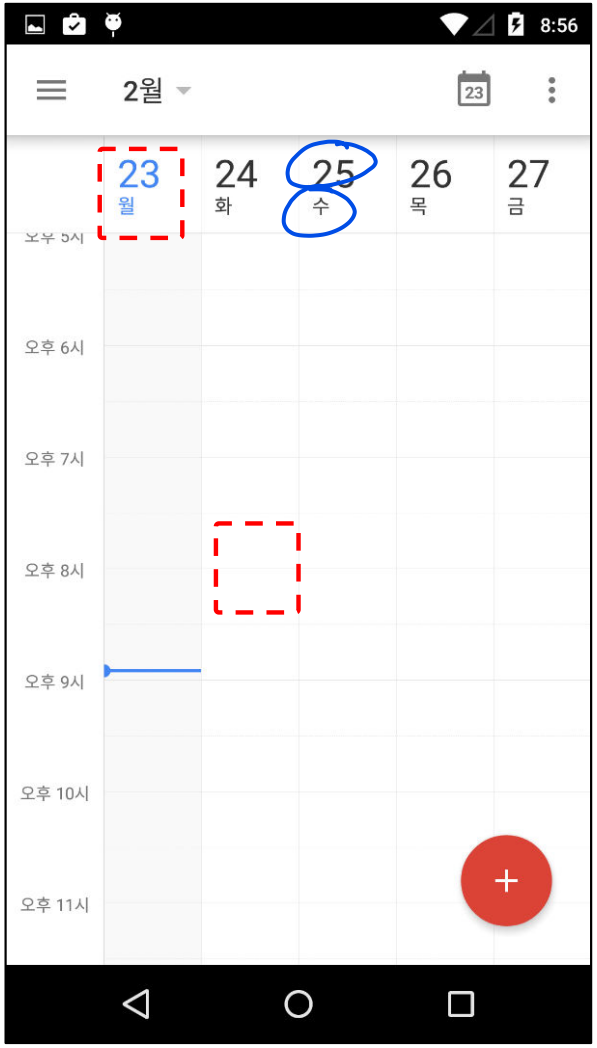
여러 개의 뷰를 보여줄 수 있는 ViewGroup의 일종



리스트 (스크롤한다)



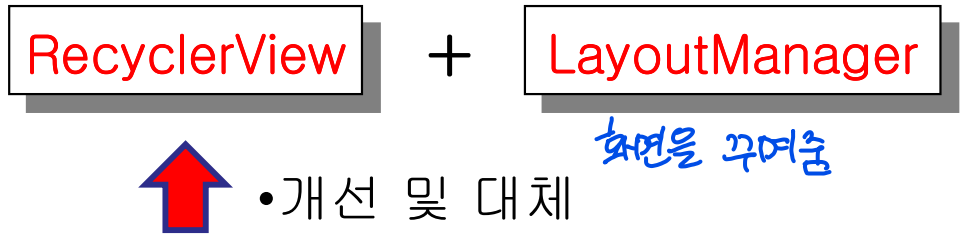
갤러리

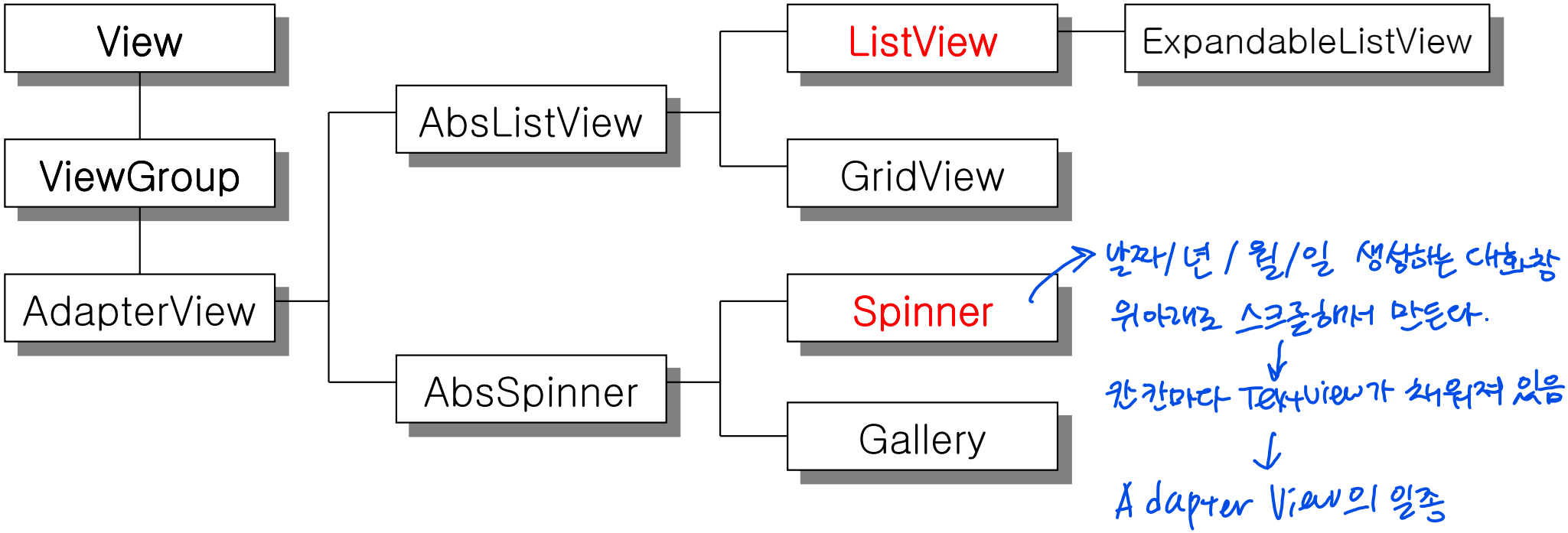


AdapterView 개요 2

- 여러 개의 View를 일정한 형태로 표시할 수 있는 ViewGroup
  - 표시할 View는 Adapter 로부터 공급 받음
- 데이터 개수 제한이 없음
  - 스크롤을 사용

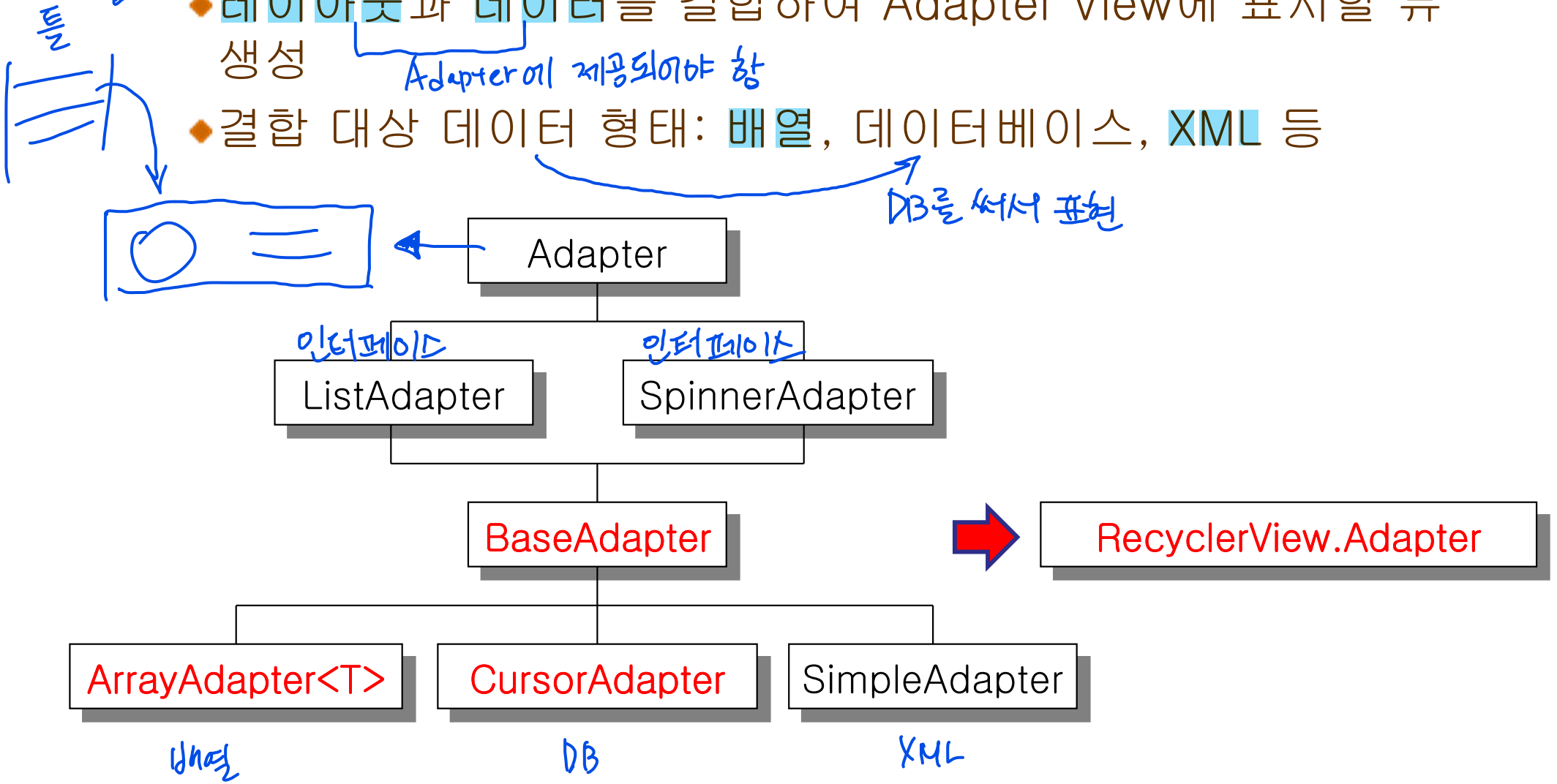
- 계층



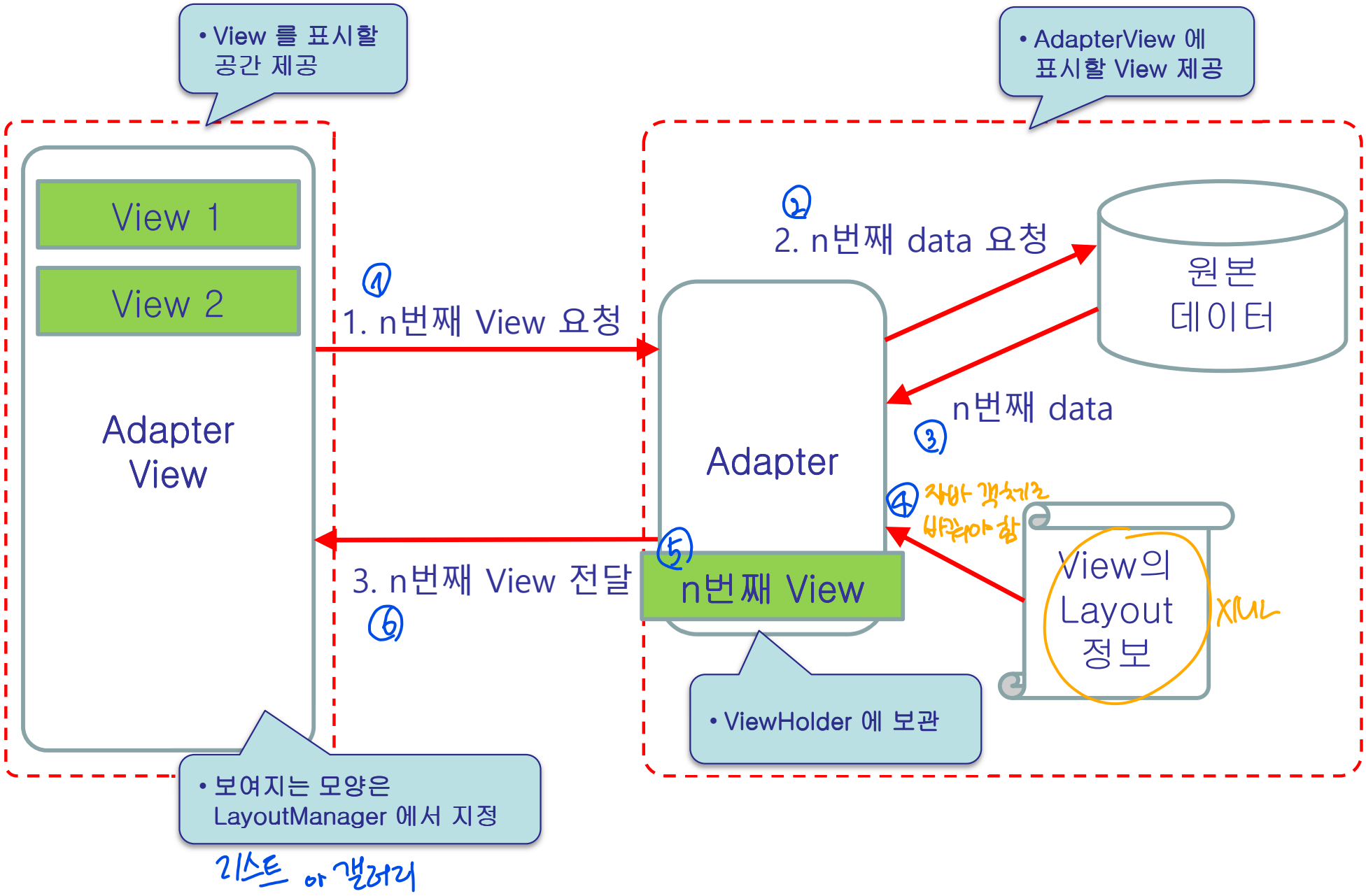


Adapter

- AdapterView에 표시할 View를 제공하는 클래스
  - 레이아웃과 데이터를 결합하여 Adapter View에 표시할 뷰 생성
  - 결합 대상 데이터 형태: 배열, 데이터베이스, XML 등



AdapterView 사용 개념도



RecyclerView

inflation : XML을 실제로 사용하기 위해 자바 객체로 바꾸는 과정

개요

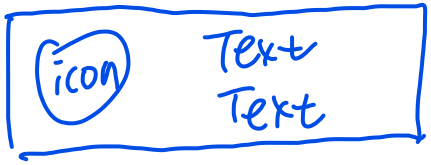
- AdapterView 의 가장 기본적인 형태의 ListView 의 성능 개선 및 다양한 표현을 위해 등장 → androidx 에 포함 (ViewHolder 없음!)
- ListView에서 직접 추가해야 하는 ViewHolder 를 기본으로 포함 (화면의 모양을 미리 보관)
  - ViewHodler 패턴 : findViewById() 호출을 최소화하여 뷰를 생성하는 효율 향상

주요 구성 요소

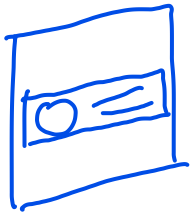
- RecyclerView (필수 – 레이아웃에 배치)
- RecyclerView.Adapter (필수 – 항목뷰 생성) → 안에 ViewHolder가 들어가 있음
- LayoutManager (필수 – 항목뷰의 배치 관리) 리스트 or 갤러리
- ItemDecorator (선택 – 항목뷰의 꾸밈)

RecyclerView 사용 절차

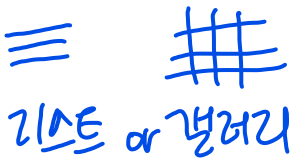
- 원본 데이터 준비
  - 배열, 데이터베이스, XML 등 다양한 형태 사용 가능
- 항목을 표현할 레이아웃 준비
  - RecyclerView 한 항목 뷰를 표현하는 레이아웃 작성



- 레이아웃에 RecyclerView 배치



- RecyclerView.Adapter 구현
  - RecyclerView.Adapter 생성 클래스 구현 및 객체 생성
  - getItemCount/onCreateViewHolder/onBindViewHolder 재정의
- RecyclerView 표시
  - LayoutManager 생성 및 지정
  - Adapter 지정 AdapterView.adapter = Adapter



1. 원본 데이터 준비

RecyclerView 에 표시할 데이터 집합

- Array, ArrayList, ArrayList<T> 등 사용

```
val dataList = ArrayList<String>()
dataList.add("모바일소프트웨어")
dataList.add("웹서비스")
dataList.add("네트워크")
dataList.add("시스템프로그래밍")
dataList.add("시스템/네트워크보안")
```

별개의 클래스로 분리하여 관리

- Activity 는 화면 관리에 집중

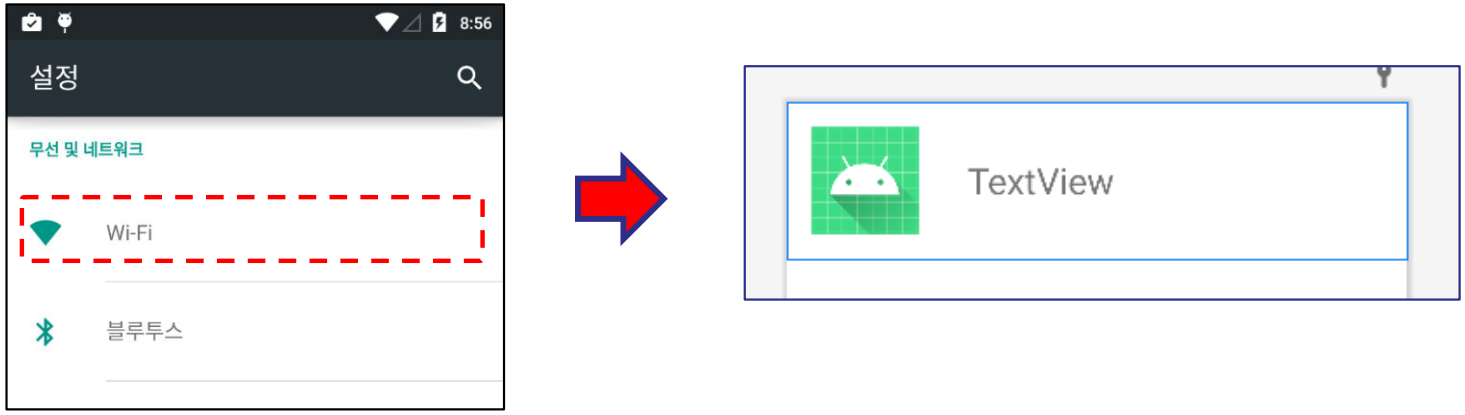
```
3   class SubjectDao() {
4       val dataList = ArrayList<String>()
5
6       init{
7           dataList.add("모바일소프트웨어")
8           dataList.add("웹서비스")
9           dataList.add("네트워크")
10          dataList.add("시스템프로그래밍")
11          dataList.add("시스템/네트워크보안")
12      }
13  }
```

• DAO (Data Access Object)
– 데이터에 접근하기 위한 용도로 사용하는 클래스 (디자인패턴 용어)

```
val dao = SubjectDao()
val dataList = dao.dataList
```

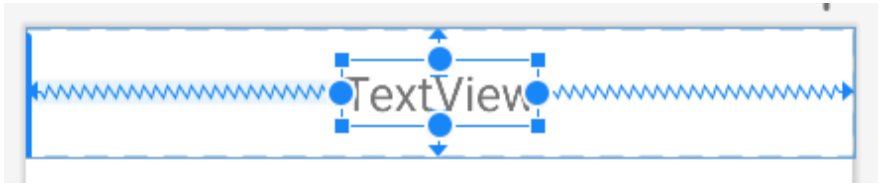
2. 항목을 표현할 레이아웃 준비

각 항목 뷰를 표현하기 위한 레이아웃의 예

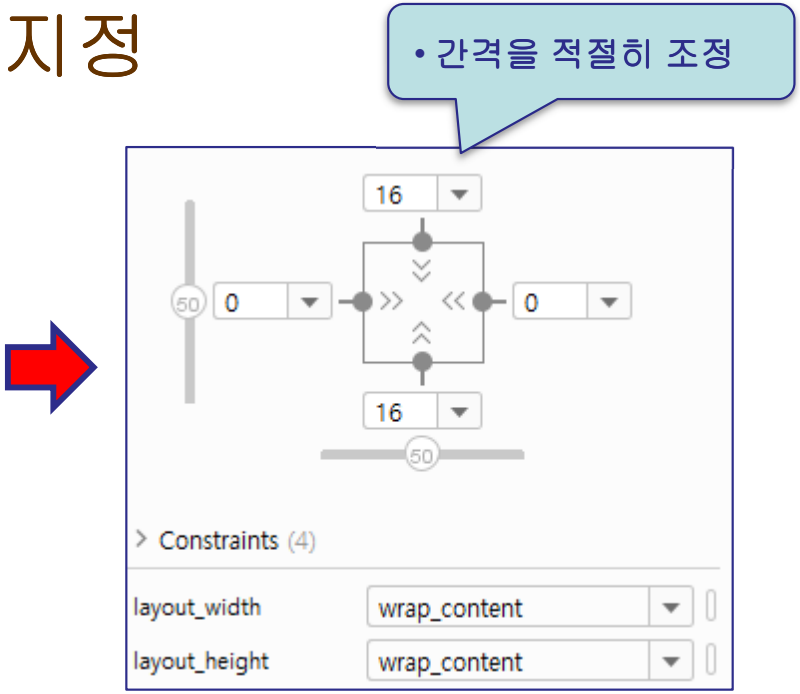


TextView 하나를 갖는 레이아웃 지정

- list_item.xml (ConstraintLayout)
  - TextView ID: tvText

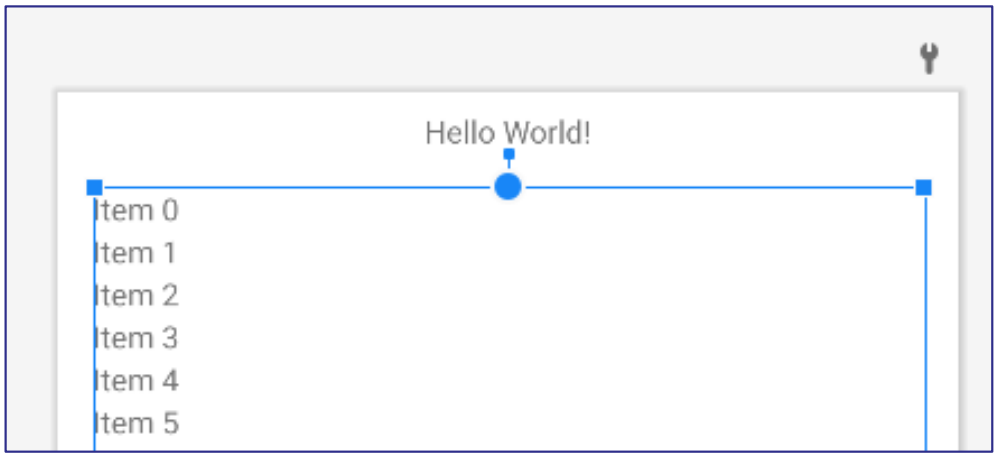


- ConstraintLayout 높이/너비
  - 높이: wrap_content 지정
  - 너비: match_parent 지정



3. 레이아웃에 RecycleView 배치

- 레이아웃에 적절히 배치
  - ID: recyclerView



- 각 항목이 보여지는 모양을 꾸미고자 할 경우 RecyclerView.ItemDecoration 사용

4. RecyclerView.Adapter의 구현

내부 클래스

개요

- RecyclerView 의 각 항목 View 를 생성 (원본 데이터와 결합)
- 뷰 생성 효율을 높이기 위해 ViewHolder 사용

•MyAdapter

```
10  class MyAdapter(val context: Context, val layout: Int, val list : ArrayList<String>)
11      : RecyclerView.Adapter<MyAdapter.MyViewHolder>() {
12
13      // RecyclerView 에 표시할 전체 뷰의 개수 == 원본 데이터의 개수
14      override fun getItemCount(): Int {
15          return list.size
16      }
17
18      // 각 항목의 뷰를 보관하는 Holder
19      override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
20          val view = LayoutInflater.from(context).inflate(layout, parent, false)
21          return MyViewHolder(view)
22      }
23
24      // Holder 에 보관중인 View 에 원본 데이터 연결
25      override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
26          holder.tvText.text = list[position]
27      }
28
29      // 항목의 뷰를 생성한 후 멤버변수로 보관하는 ViewHolder
30      class MyViewHolder(view: View) : RecyclerView.ViewHolder(view) {
31          val tvText : TextView = view.findViewById(R.id.tvText)
32      }
33  }
```

전체 항목 뷰의 개수 확인

항목을 표시하는 layout 생성 후 ViewHolder에 보관

항목 뷰의 내용과 원본 데이터 연결

화면상의 순서 == 원본데이터 순서

Adapter 내부에 ViewHolder 클래스 추가

항목 View 를 구성하는 내부 View 보관

5. RecyclerView 표시

LayoutManager 지정

- RecyclerView 내부에 각 항목 View 가 어떠한 형식으로 보여질지 지정
- Linear/Grid/StaggeredGrid LayoutManager

•MainActivity

```
12    override fun onCreate(savedInstanceState: Bundle?) {
13        /* setContentView() 관련 코드 생략 */
14
15        val dao = SubjectDao()
16        val dataList = dao.dataList
17
18        val layoutManager = LinearLayoutManager(this)
19        layoutManager.orientation = LinearLayoutManager.VERTICAL    // 생략 가능
20        binding.recyclerView.layoutManager = layoutManager
21
22        val adapter = MyAdapter(this, R.layout.list_view, dataList)
23        binding.recyclerView.adapter = adapter
24    }
```

• 원본 데이터 준비

• LayoutManager 지정

• Apdater 생성 및 RecyclerView 에 연결

실습

- 다음과 같은 리스트 뷰를 갖는 앱을 생성하시오.
  - 내용은 자유로이 구성
  - 우선 Activity에 데이터를 생성하여 작성
  - 위의 실습 완료 후 DAO 를 별도로 작성

이벤트 처리

이벤트 핸들러 작성

- 항목 View 또는 내부에 포함한 뷰에 이벤트 핸들러 연결
- ViewHolder 생성 시 이벤트 핸들러 추가

•MyAdapter 내부의 ViewHolder

```
// 항목의 뷰를 생성한 후 멤버변수로 보관하는 ViewHolder
class MyViewHolder(view: View) : RecyclerView.ViewHolder(view) {
    val tvText : TextView = view.findViewById(R.id.tvText)
    init {
        view.setOnClickListener{ it: View!
            Toast.makeText(view.context, "항목 $adapterPosition View 터치!", Toast.LENGTH_SHORT).show()
        }
        tvText.setOnClickListener{ it: View!
            Toast.makeText(view.context, "TextView Click!", Toast.LENGTH_SHORT).show()
        }
    }
}
```

• 항목을 표현하는 view 에 이벤트 핸들러 구현

• 항목 iew 내부의 뷰에 이벤트 핸들러 구현

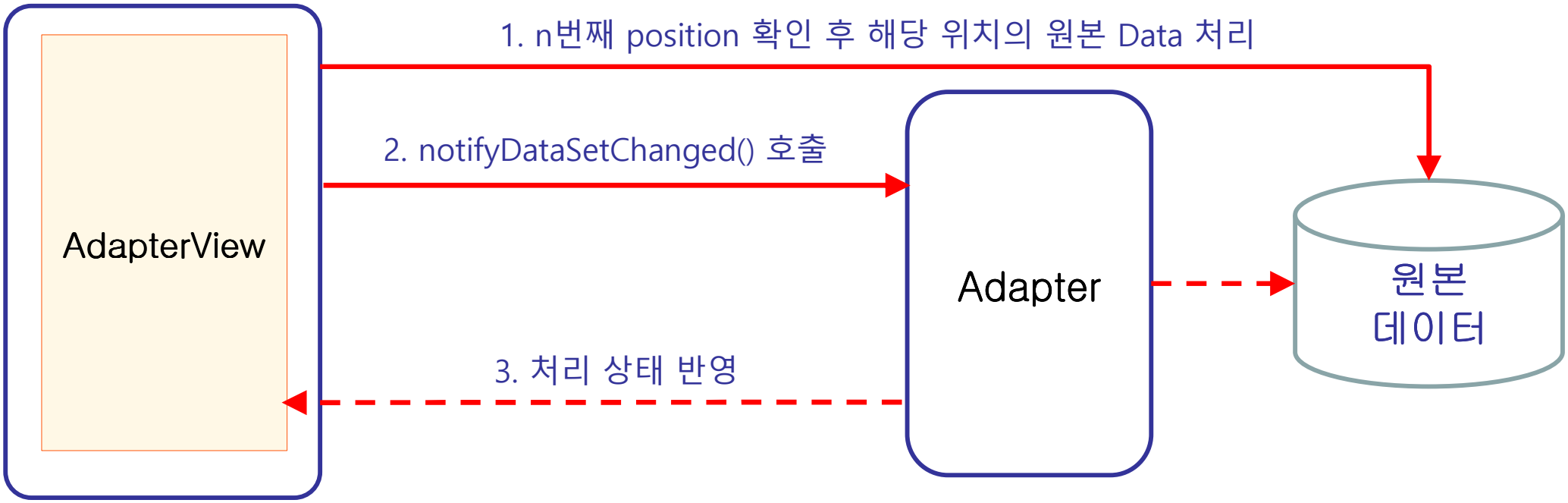
- adapterPosition : 현재 선택한 항목의 index 를 확인할 수 있는 ViewHolder 의 멤버변수 (getAdapterPosition())

항목 편집 처리

Adapter View의 항목 처리 절차

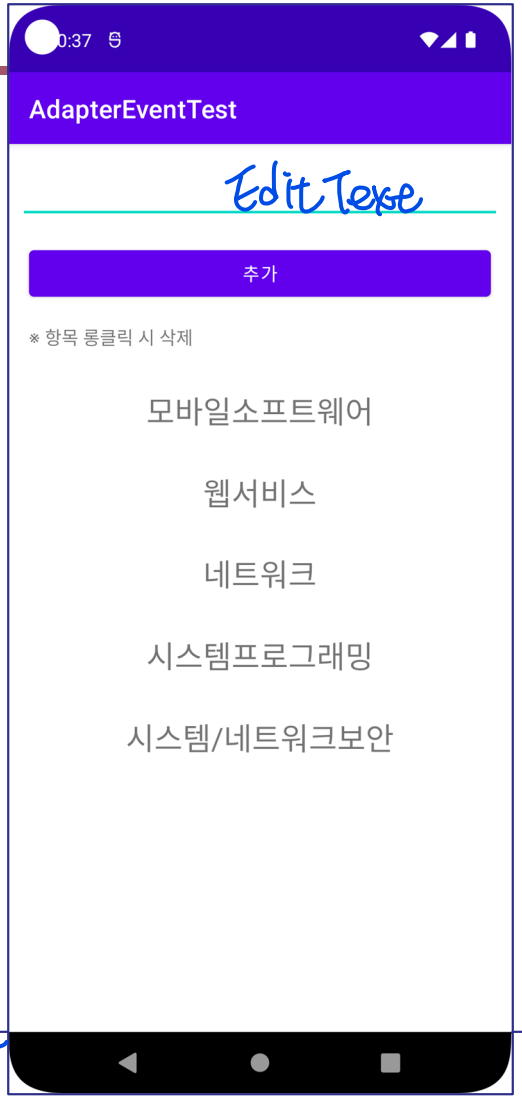
- 처리할 항목의 위치 확인 → position (adapter Position)
- 원본 데이터에서 position에 해당하는 원본 데이터 처리 → 데이터의 수정/삭제
- 원본 데이터를 사용하는 어댑터에 변경 알림

• RecyclerView.Adapter.notifyDataSetChanged()



실습

- RecyclerViewTest 프로젝트를 사용
  - 항목의 내용은 과목명이 아닌 다른 종류로 변경 (음식명 등)



- 다음과 같은 기능을 완성
  - EditText에 문자열 입력 후 [추가] 버튼을 눌러 항목 추가
  - RecyclerView 롱클릭 시 해당 항목 삭제

- 참고 – ViewHolder 수정
  - 생성자의 매개변수로 adapter 추가

inner class 사용해면 됨!

```
// 항목의 뷰를 생성한 후 멤버변수로 보관하는 ViewHolder
class MyViewHolder(adapter: MyAdapter, view : View) : RecyclerView.ViewHolder(view) {
    val tvText : TextView = view.findViewById(R.id.tvText)
    init {
        // adapter의 notifyDatasetChanged 및 adapter 의 멤버변수 list 사용하여 이벤트 처리
    }
}
```