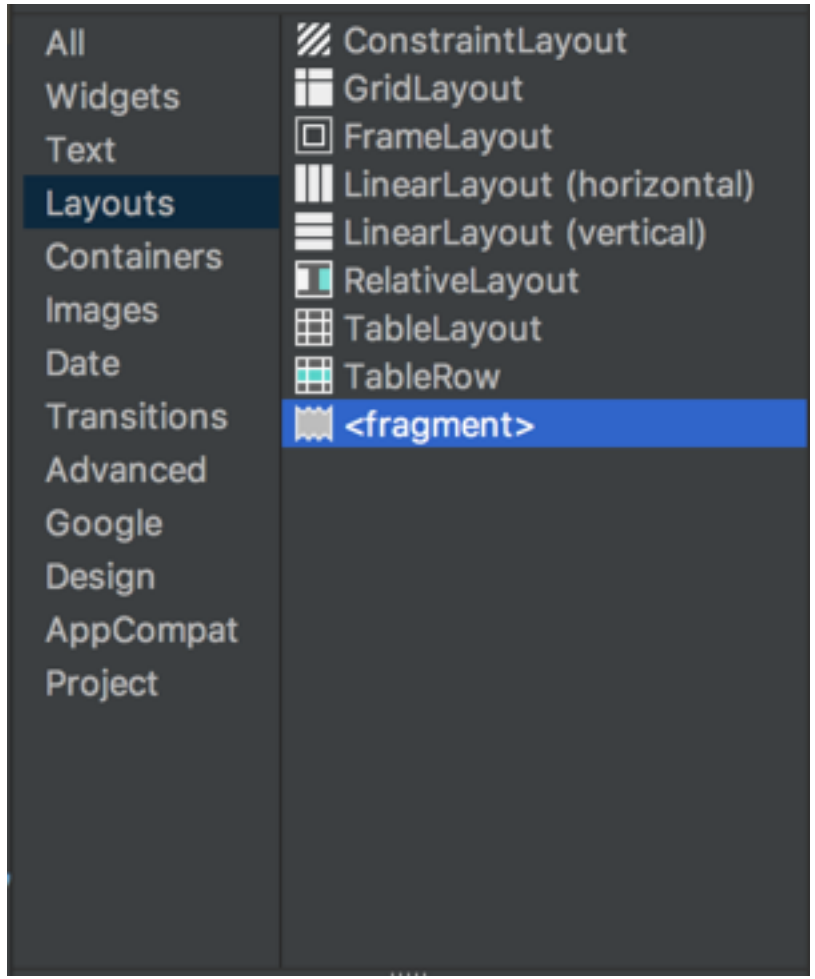
MASH UP

CONTENTS

MASH UP

1. Layout

01 Layout _01 Layout?



- UI에 대한 시각적 구조를 정의

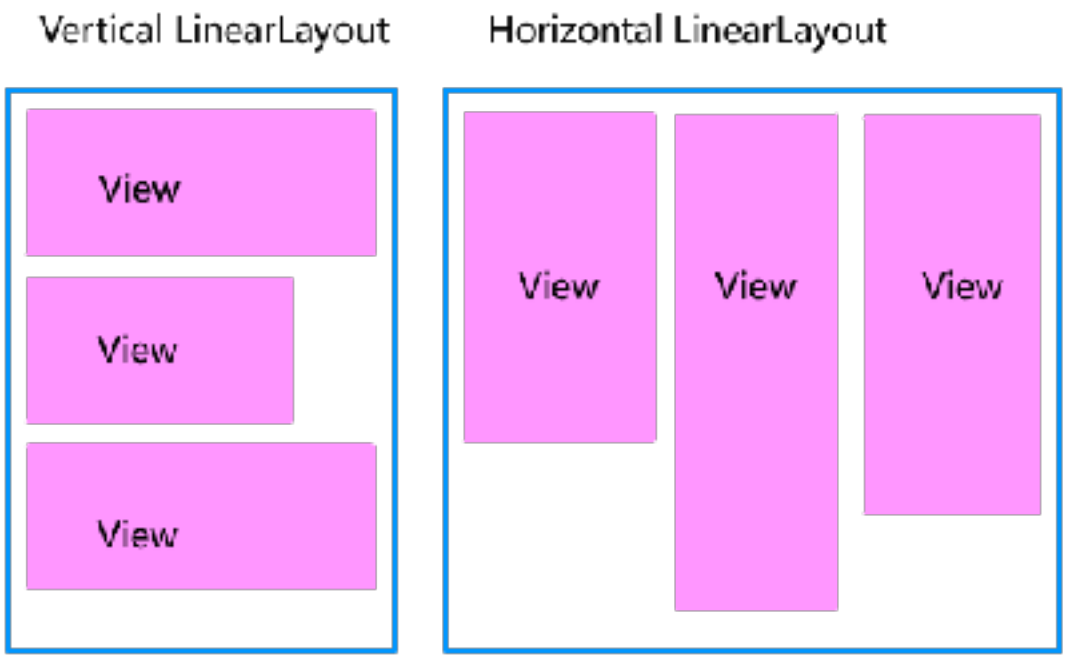
- 액티비티 또는 앱 위젯에 대한 UI가 이에 해당
Android는 앱의 UI를 선언하고 관리하기 위한 메소드를 제공

- UI를 XML로 선언하며 앱을 동작을 제어하는 코드로부터 별도로 표시 가능

- 런타임에 View 객체를 인스턴스화하는 것에 흥미가 있는 경우, ViewGroup 및 View 클래스를 참조

https://developer.android.com/guide/topics/ui/declaring-layout.html?hl=ko#write

01 Layout _02 LinearLayout



- LinearLayout은 세로 또는 가로의 단일 방향으로
  모든 하위 항목을 정렬하는 뷰 그룹

- android:orientation 특성을 사용하여 레이아웃 방향을 지정

- LinearLayout의 모든 하위 항목은 순차적으로 스택
  - 세로 목록의 경우 너비에 상관없이 한 행당 하나의 하위 항목
  - 가로 목록의 경우 높이가 한 행 높이

- LinearLayout은 하위 항목 사이의 여백 및 각 하위 항목의 중력
  - (오른쪽, 가운데, 왼쪽 정렬)을 유지

- View가 계층적으로 유지
  - 하나의 View를 찾기 위해서 모든 View에 접근해야함

01 Layout _02 LinearLayout

android:layout_weight 속성

- 이 특성은 뷰가 화면에서 얼마나 많은 공간을 차지해야 하는지와 관련하여 뷰에 '중요한' 값을 할당

- 큰 가중치 값을 사용하면 상위 뷰(LinearLayout)에서 남은 공간을 모두 채우도록 확장

- 하위 뷰에 가중치 값을 지정

- 세 개의 텍스트 필드 존재
    - 그중 두 개를 가중치 1로 선언하고 다른 하나에는 가중치가 지정 안함
        - 가중치가 지정되지 않은 세 번째 텍스트 필드는 확장되지 않고 해당 콘텐츠에 필요한 영역만 차지
        - 다른 두 텍스트 필드는 세 필드가 모두 측정된 후 남은 공간을 균등하게 채우도록 확장
    - 세 번째 필드에 가중치가 2로 지정되면 다른 두 필드보다 더 중요한 것으로 선언

01 Layout _02 RelativeLayout

- RelativeLayout은 하위 뷰를 상대 위치에 표시하는 레아이웃
각 뷰의 위치는 각 뷰의 위치를 기준(다른 뷰의 왼쪽 또는 아래로)

- 부모 RelativeLayout 영역을 기준(예 : 아래쪽, 왼쪽 또는 가운데에 정렬)

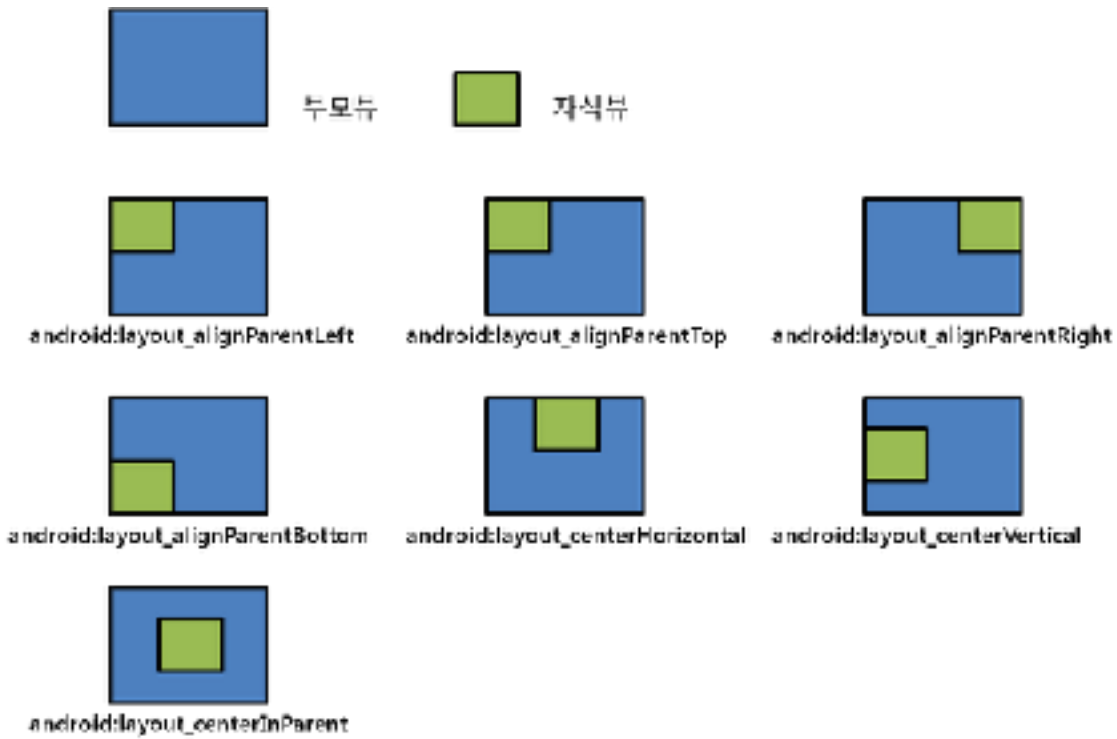
참고 : 성능 및 툴링 지원을 향상 시키려면 대신 ConstraintLayout을 사용

- RelativeLayout은 중첩 된 뷰 그룹을 제거하고 레이아웃 계층을 평평하게 유지하여 성능을 향상

- 사용자 인터페이스를 디자인하는 데 매우 유용한 유틸리티

- 여러 개의 중첩 된 LinearLayout을 사용하는 경우 하나의 RelativeLayout 으로 대체

01 Layout _02 RelativeLayout



layout_above ~의 위에 배치하라
layout_below ~의 밑에 배치하라
layout_toLeftOf ~의 왼쪽에 배치하라
layout_toRightOf ~의 오른쪽에 배치하라
---
layout_alignTop ~와 위쪽 변을 맞춰라
layout_alignBottom ~와 밑쪽 변을 맞춰라
layout_alignLeft ~와 왼쪽 변을 맞춰라
layout_alignRight ~와 오른쪽 변을 맞춰라
---
layout_alignParentTop true이면 부모와 위쪽 변을 맞춰라
layout_alignParentBottom true이면 부모와 밑쪽 변을 맞춰라
layout_alignParentLeft true이면 부모와 왼쪽 변을 맞춰라
layout_alignParentRight true이면 부모와 오른쪽 변을 맞춰라
---
layout_centerHorizontal true이면 부모의 수평 중앙에 배치하라
layout_centerVertical true이면 부모의 수직 중앙에 배치하라
layout_centerInParent true이면 부모의 수평, 수직 중앙에 배치하라

01 Layout _02 FrameLayout

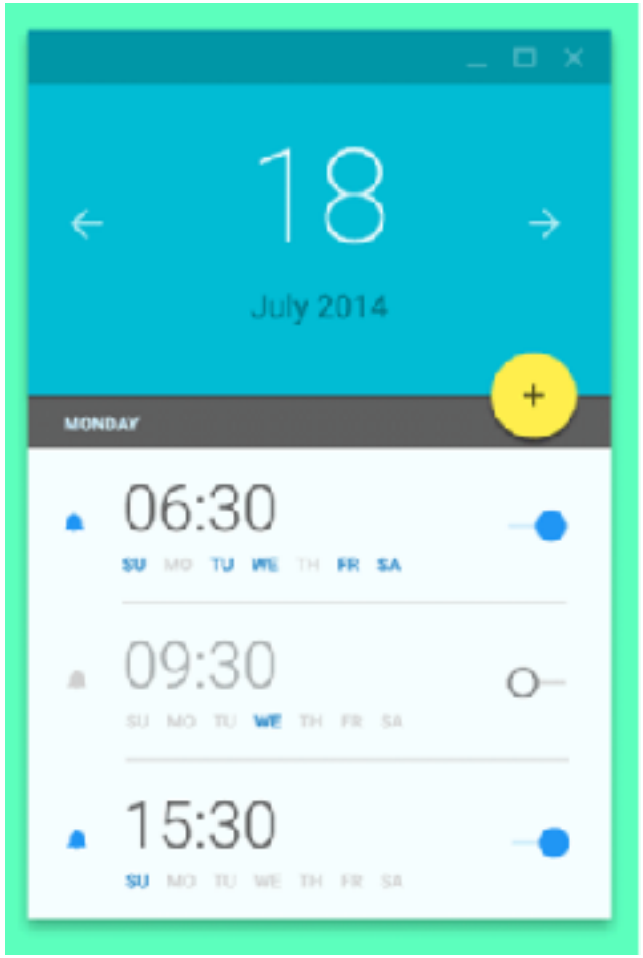
- FrameLayout은 단일 항목을 표시하기 위해 화면의 영역을 차단하도록 설계
    - 일반적으로 FrameLayout은 하나의 하위 뷰를 유지하는 데 사용

- FrameLayout에 여러 자식 뷰를 추가하려면 자식뷰에 android:layout_gravity 특성을 사용
    - 각 자식 뷰를 FrameLayout 내의 위치를 제어

- 하위 뷰는 스택에 그려지고 가장 최근에 추가 된 뷰가 맨 위에 표시

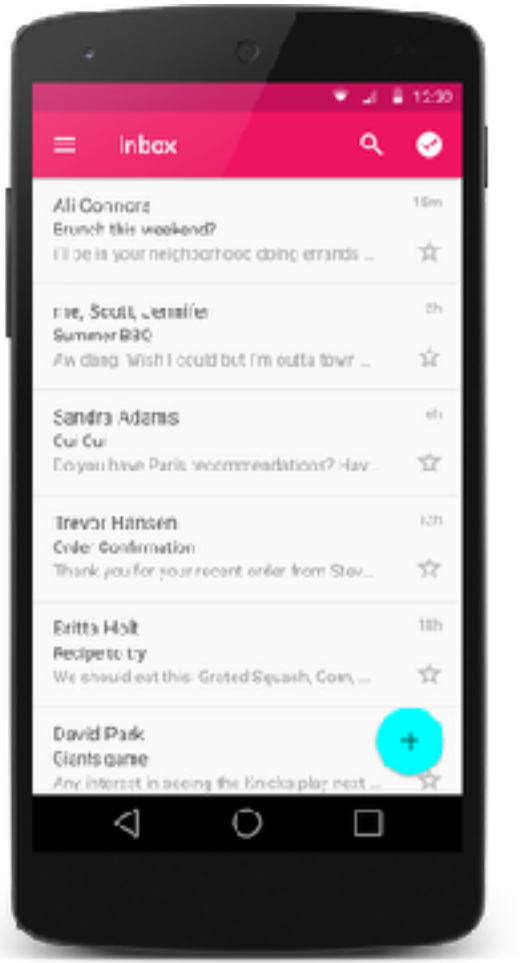
- FrameLayout의 사이즈는 가장 큰 자식 뷰의 크기(match_parent가 아닌 경우)

01 Layout


18
July 2014
MONDAY
06:30
SU MO TU WE TH FR SA
09:30
SU MO TU WE TH FR SA
15:30
SU MO TU WE TH FR SA

2. Recycler View

02 RecyclerView _01 Recycler View?

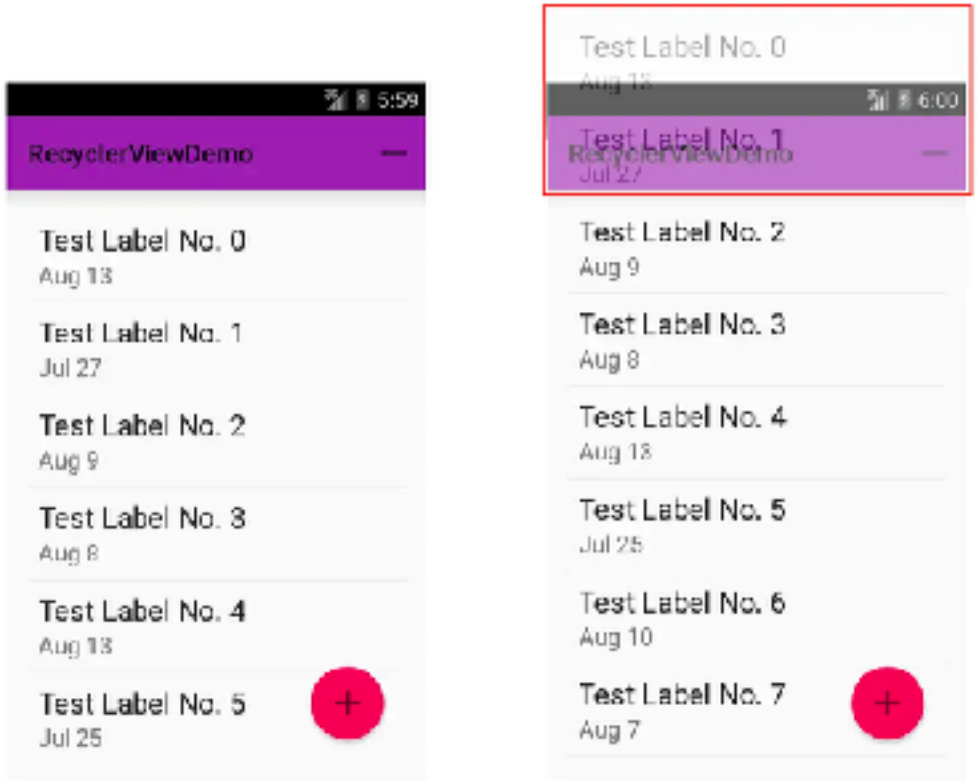


- RecyclerView 위젯은 ListView의 더욱 향상되고 유연해진 뷰

- 한정된 수의 뷰를 유지함으로써 효율적으로 스크롤할 수 있는 큰 **데이터 집합**을 표시하기 위한 컨테이너

https://developer.android.com/topic/libraries/support-library/features.html

02 RecyclerView _02 재사용?



- 안드로이드는 기본적으로 한정된 자원을 지님

- 각 요소들은 일정량의 메모리 자원을 소모

- 300개의 데이터 집합을 표현
  - 화면에 보여지는 6개의 데이터 집합의 공간을 생성
  - 해당 공간을 재활용하여 나머지 데이터 집합을 표현

02 RecyclerView _03 ListView vs RecyclerView

ListView는 커스텀하기에는 구조적 문제로 많은 제약과, 성능 문제를 가짐

RecyclerView는 문제를 해결하기 위해 다양한 형태로 유연하며 성능에 중점을 두어 만듬

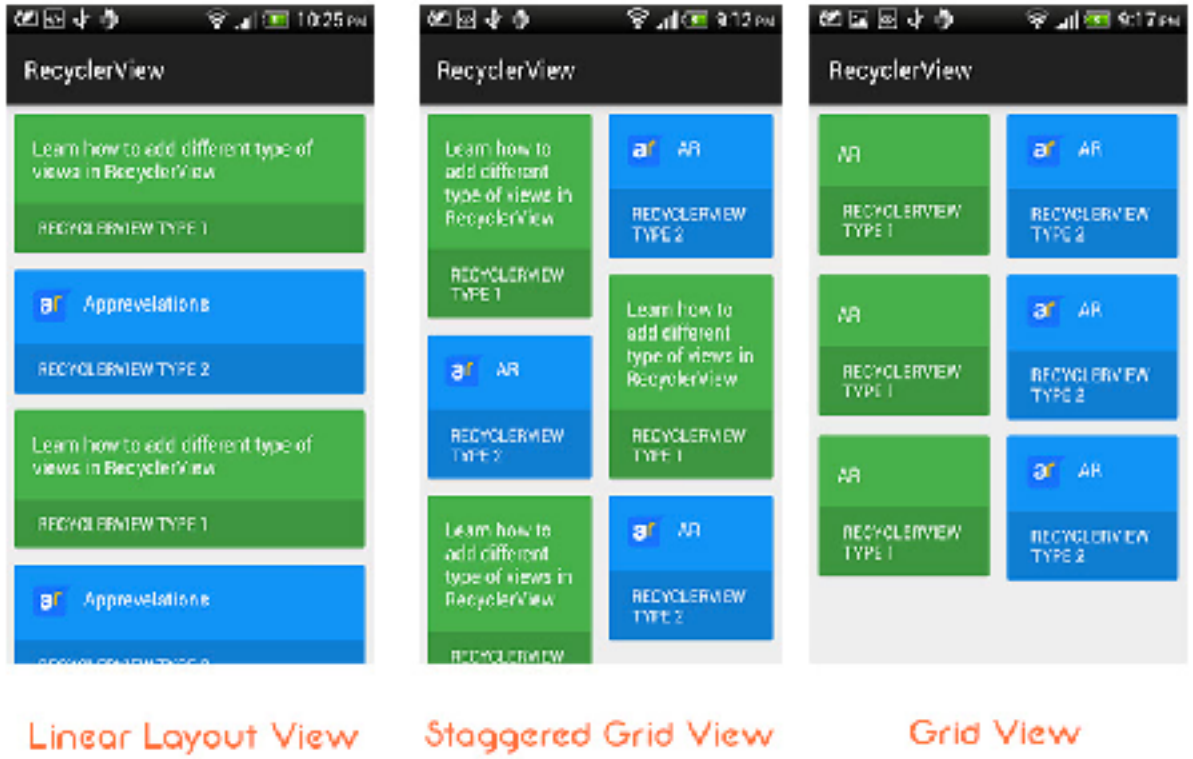
LayoutManager – 아이템의 항목을 배치

ViewHolder – 재활용 View에 대한 모든 서브 뷰를 보유

ItemDecoration – 아이템 항목에서 서브뷰에 대한 처리

ItemAnimation – 아이템 항목이 추가, 제거되거나 정렬될때 애니메이션 처리

02 RecyclerView _04 LayoutManager



Linear Layout View Staggered Grid View Grid View

- RecyclerView를 생성 시 반드시 **필요**
- 모든 아이템의 뷰 레이아웃을 관리
- 기본적으로 제공하는 LayoutManager
  - LinearLayoutManager
    - 수평/수직의 스크롤 리스트
  - GridLayoutManager
    - 그리드 리스트
  - StaggeredGridLayoutManage
    - 높이가 불구칙적인 형태의 그리드 리스트

```
LinearLayoutManager layoutManager = new LinearLayoutManager(context);
layoutManager.setOrientation(LinearLayoutManager.VERTICAL);
layoutManager.scrollToPosition(currPos);
recyclerView.setLayoutManager(layoutManager);
```

02 RecyclerView _05 ViewHolder

ListView에서 많이 사용된 패턴, 강제적으로 제한하지 않음

RecyclerView에서는 ViewHolder를 필수로 사용하는 구조로 바뀜

View.findViewById(Int)는 코스트가 높은 메소드

그렇기 때문에 ViewHolder 패턴을 사용하여 View를 캐싱

```
final static class ListItemViewHolder extends RecyclerView.ViewHolder {
    TextView label;
    TextView dateTime;

    public ListItemViewHolder(View itemView) {
        super(itemView);
        label = itemView.findViewById(R.id.txt_label_item);
        dateTime = itemView.findViewById(R.id.txt_date_time);
    }
}
```

02 RecyclerView _05 Adapter

- 아이템의 데이터와 뷰 간의 처리를 한다.

- 아래 3가지의 메소드를 오버라이드

```
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
```

ViewType에 해당하는 ViewHolder를 생성하여 Return
ViewHolder는 각 Item View를 캐싱하고 있음
그러므로 Adapter는 ViewHolder를 이용해 item view를 관리

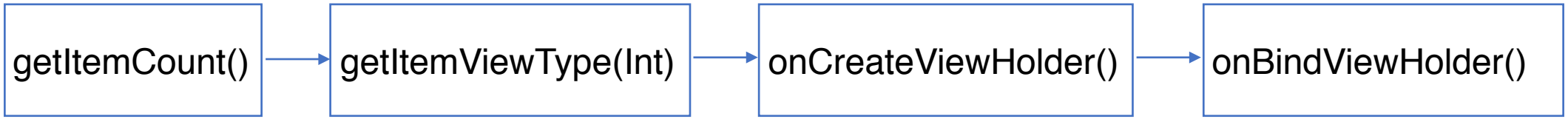
```
public void onBindViewHolder(@NonNull RecyclerView.ViewHolder holder, int position)
```

만들어진 ViewHolder에 데이터를 넣는 작업
현재 position에 맞는 데이터를 ViewHolder가 관리하는 View들에 Binding

```
public int getItemCount()
```

표현할 데이터의 개수

02 RecyclerView _05 Adapter



폰 화면에 10개의 list가 보인다면,

맨처음 getItemCount가 불리고,

그 후 10번 getItemViewType, onCreateViewHolder, onBindViewHolder가 연속적으로 호출

02 RecyclerView

https://www.raywenderlich.com/170075/android-recyclerview-tutorial-kotlin

https://www.androidhive.info/2016/01/android-working-with-recycler-view/