[WorksMobile Intern Project]

[개인 보고서]

[Mobile Dev1 성지훈]

The table of contents

1. 개요

현재 사내 WORKS 및 네이버웍스 애플리케이션에 사용되는 구성원 일정 뷰는 PC 구성원 일정 뷰와 달리 구성원들의 일정을 한눈에 보기 어렵다는 단점과 구성원 일정을 일간 단위로만 볼 수 있다는 점을 개선하고 추후 다른 사용자도 쉽게 사용할 수 있도록 구성원 일정뷰를 커스텀뷰로 제작하여 오픈소스 라이브러리로 제작하는 것을 목표로 하며 라이브러리에 대한 사용방법 및 샘플 애플리케이션을 제작하여 사용자에게 제공하는 것을 최종 목표로 한다.

2. 구조

2.1 라이브러리 구조

라이브러리에 사용되는 커스텀뷰는 Vertical LinearLayout 으로 구성되어 있으며 내부에 FrameLayout 이 있다. FrameLayout 을 사용한 이유는 구성원 일정을 그리기 위한 mainRecyclerView 위에 이름만을 표기하는 nameRecyclerView 를 겹쳐서 놓아야 하기 때문이다.
이처럼 리사이클러뷰를 두가지로 나누어 사용하는 이유는 커스텀뷰가 상하좌우 스크롤이 가능해야 하며 모바일 화면이 PC 에 비해 작은 것을 고려하여 좌우 스크롤시 좌측 헤더가 따라서 스크롤 되어 화면에서 사라지는 것을 방지하고 좌측헤더를 고정시켜야 하기 때문이다. 또한 구성원 일정뷰는 Vertical 한 RecyclerView 이므로 좌우로 스크롤이 불가능하다. 따라서 mainRecyclerView 를 HorizontalScrollView 로 감싸는 형태로 구성하였다.

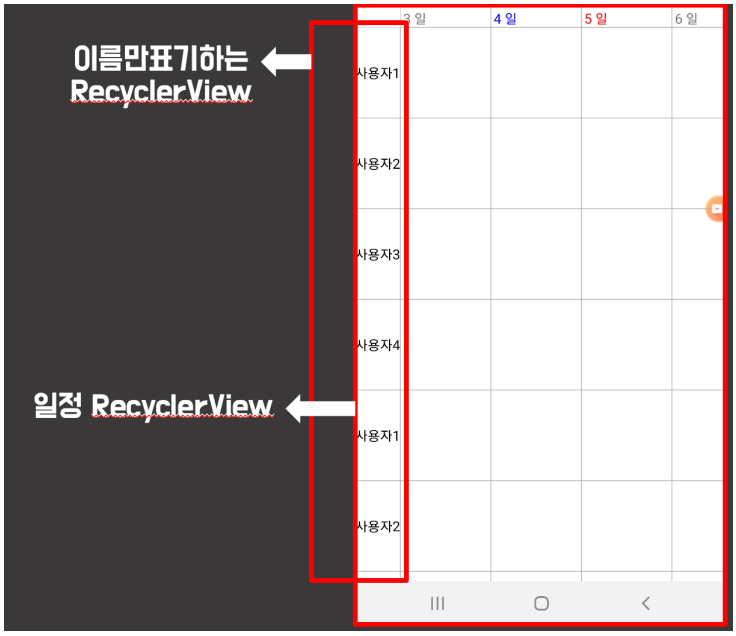

그림 1

그림 1 에서 좌측 이름만 표기하는 RecyclerView 는 구성원 일정뷰를 표기하는 RecyclerView 가 좌우로 스크롤되어도 다른 뷰이기 때문에 항상 고정되어 있는 상태이다. 실제로 좌측 헤더를 따로 고정시킨 것은 아니지만 사용자가 볼 때 불편함이 없도록 아이디어를 생각하여 구현한 방법이다. 또한 상단 헤더 고정을 위해 mainRecyclerView 에서 ItemDecoration 을 추가하여 상단 헤더를 기존 RecyclerView 위에 다시 그려 상단헤더가 고정되어 있는 것 처럼 보이도록 구현하였다. 따라서 좌우 스크롤 시 좌측 사용자의 이름헤더는 고정되어 있으며 상하 스크롤 시 시간 헤더는 고정되어 있는 형태이다. 아래는 커스텀뷰의 구조이다.

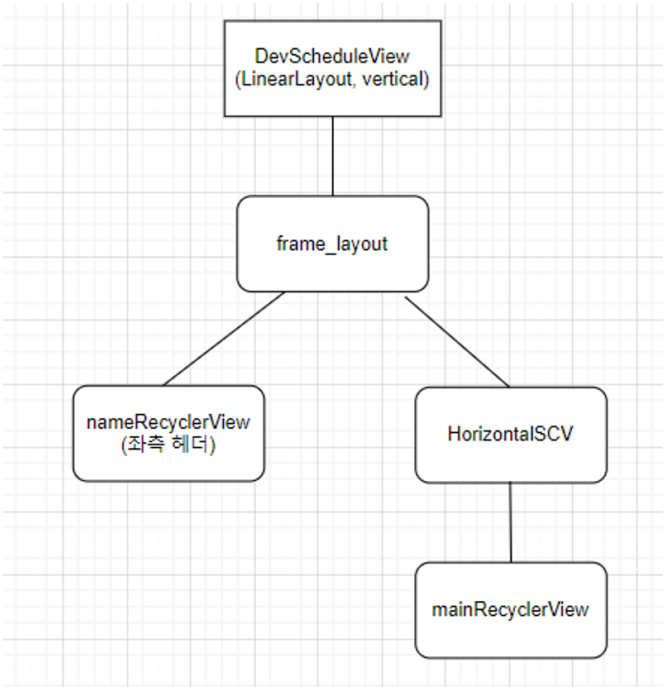

그림 2

3. 구현과정

3.1 Library

라이브러리는 크게 mainview, mainrv, leftrv, interface, model패키지로 구분되어 있으며 공통적으로 사용할 메소드는 singleton object로 Utils.kt에 구현되어 있다.

3.1.1 DevScheduleView

mainview 에는 DevScheduleView 인 메인 커스텀뷰가 구현되어 있다. 커스텀뷰인 DevScheduleView 는 Vertical LinearLayout 이다.

```
if(attrs!=null) {
    val a : TypedArray = context.obtainStyledAttributes(attrs, R.styleable.DevScheduleView)
    view_type = a.getInt(R.styleable.DevScheduleView_viewType, defValue: 0)
    per_topheaderwidth = a.getDimensionPixelSize(R.styleable.DevScheduleView_per_topheaderwidth,getDP( value: 50))
    per_leftheaderheight = a.getDimensionPixelSize(R.styleable.DevScheduleView_per_leftheaderheight,getDP( value: 50))
    a.recycle()
}
```

먼저 사용자가 xml 에서 커스텀뷰를 추가하여 사용할 경우, 사용자는 viewType 과 per_topheaderwidth, per_leftheaderheight 총 3 가지의 attribute 를 설정할 수 있다. viewType 은 사용자가 커스텀뷰를 일간/주간으로 설정할 수 있고, per_topheaderwidth 와 per_leftheaderheight 는 각각 상단 헤더의 너비와 좌측 헤더의 높이의 크기를 조절할 수 있다. 아래 그림에 빨간색 영역이 각 attribute 의 지정값이다. getDP()의 경우 사용자가 입력한 Integer 값을 DP 단위로 변환시켜주는 메소드이다.

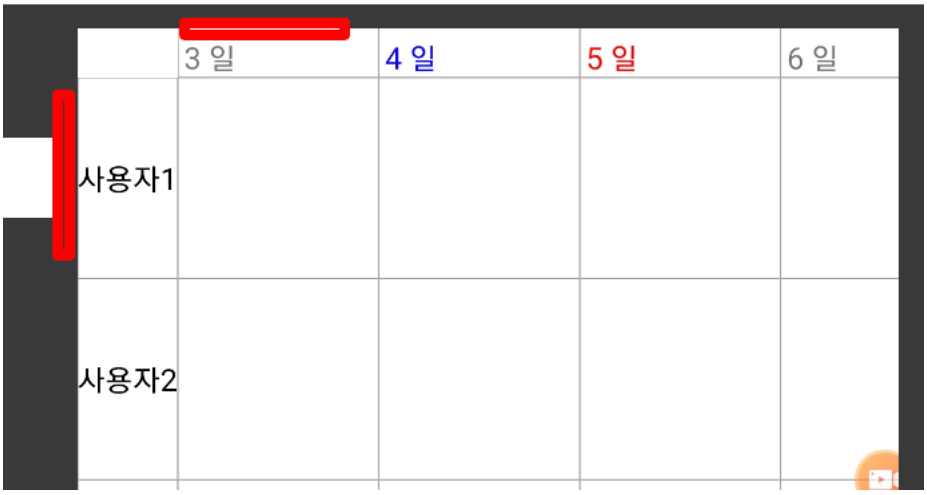

그림 3

현재 그림 3 은 주간 뷰에서 사용하는 모습이다. 사용자가 viewType 을 1 로 설정한 모습이며 viewType 이 0 인경우에는 그림 4 와 같이 시간 헤더가 별도로 표기된다.

그림 4

addAnimator() : 모바일 화면에서는 PC 구성원 일정뷰의 주간 일정처럼 일정을 쌓아가는 형태로 구현하는 경우, 일정의 개수가 최대 2~3 개까지만 볼 수 있다는 점을 감안하여 이를 해결하기 위해 ValueAnimator 를 사용하였다. 좌측 헤더 클릭시 slideAnimator 를 사용하여 좌측 헤더와 해당 포지션의 메인 리사이클러뷰 아이템의 높이를 300DP 로 늘리도록 구현하였으며 cancelAnimator 의 경우 사용자가 헤더를 한번더 클릭하거나 다른 사용자 이름을 클릭할 경우 기존의 증가했던 높이를 다시 되돌리는 ValueAnimator 이다.

createViewForSanpping() : 좌우 스크롤 시, 사용자가 지정한 topheaderwidth 에 맞추어 스크롤 하기 위해서 사용자가 지정한 정확한 width 만큼의 스크롤 값을 측정해야한다. 이때 측정되는 width 값을 측정하기 위해 지정한 크기에 맞는 View 를 생성하고 이 뷰(getWidthView)의 width 값을 측정하여 좌우 스크롤시 칸 단위로 스크롤 될 수 있도록 Snapping 을 제어한다.

addBaseParams() : 커스텀뷰의 기본적인 파라미터를 설정하고 커스텀뷰 내에 FrameLayout 을 추가한다.

addFrameParams() : 프레임 레이아웃의 파라미터를 설정하고 getWidthView 를 프레임 레이아웃에 추가한다.

addCoverHorizontalView() : 메인 리사이클러뷰를 감싸기 위한 HorizontalScrollView 의 기본 파라미터를 설정하고 프레임 레이아웃 내에 추가한다.

addMainRecyclerView() : mainRecyclerView 를 생성하고 리사이클러뷰의 어댑터 설정 및 리사이클러뷰를 설정한다. mainRecyclerView 는 LinearLayoutManager 를 사용하며 상단 헤더

고정을 위해 ItemDecoration 을 사용하고, 리사이클러뷰의 크기를 고정하는 setHasFixedSize(true)와 어댑터에서 onBindViewHolder 를 반복적으로 호출하지 않도록 setItemViewCacheSize()를 사용하고 리사이클러뷰를 cover_mainrv_horizontal_view 에 추가한다.

addLeftHeaderrv() : 좌측 헤더를 위한 리사이클러뷰를 생성하고 파라미터를 설정한다. 리사이클러뷰의 어댑터 설정 및 리사이클러뷰를 설정하며 메인 리사이클러뷰와 마찬가지로 setHasFixedSize(true)와 setItemViewCacheSize()를 사용한다. 다만 추후 구현할 상단 헤더 클릭 리스너를 구현하기 위해 상단 헤더에 투명한 뷰를 추가하여 사용한다.

getSectionCallback() : 메인 리사이클러뷰의 상단 헤더 고정을 위해 사용하는 StickyItemDecoration 을 위해 사용하는 콜백함수를 return 하는 메소드이다

observeScroll() : 두개의 리사이클러뷰는 상,하 스크롤 시 함께 움직여야 하므로 scrollListenerFromMainRv 라는 OnScrollListener 를 생성하고 mainRecyclerView 가 상,하로 스크롤 되는 경우 nameRecyclerView 도 함께 스크롤 되도록 nameRecyclerView.scrollBy()를 이용하여 스크롤한다. 생성된 리스너는 onDetachFromWindow 에서 제거된다. 또한 nameRecyclerView 에 ItemTouchListener 를 추가하여 터치이벤트를 제어한다. 사용자가 입력한 터치이벤트가 Action_Down 일 경우에만 이벤트를 전달하고 그 이외의 이벤트는 Intercept 한다. 따라서 nameRecyclerView 를 직접적으로 스크롤 하는 것을 방지하고 터치이벤트만 가능하도록 구현하였다.
또한 이름헤더 클릭시 좌측 헤더의 높이를 증가시켜 일정을 볼 수 있도록 하는 SlideAnimator 는 사용자가 스크롤 할 시 자동으로 취소되도록 사용자가 mainRecyclerView 를 스크롤 할 경우 cancelExpandAnimation 을 실행한다.

snappingHorizontalScroll() : HorizontalScrollView 에서 ActionUP 또는 ActionCancel 이벤트가 발생할 때 스크롤위치를 재조정 하여 한칸 단위로 스크롤 되는 효과를 설정한다. 먼저 scrollX 값을 받아오고, 사용자가 설정한 한 칸의 width 값을 가져온다. 이 값을 이용하여 스크롤의 위치를 재조정하여 한칸씩 스크롤 되는 효과를 준다.

setOnClickScheduleListener(),
setOnClickTimeHeaderListener, setOnClickNameheaderListener() : 사용자가 외부에서 리스너를 별도로 설정하고자 할 때 리스너를 등록할 수 있는 메소드이다. 각각 스케줄클릭, 상단 시간헤더 클릭, 좌측 이름헤더 클릭 시 동작을 정의할 수 있다.

setDate() : 사용자가 외부에서 날짜를 설정할 수 있다. Time 객체를 파라미터로 받으며 커스텀뷰 생성 이후 필수로 날짜를 설정해야 한다.

submitUserList() : List<User>객체를 파라미터로 받으며 사용자가 일정 또는 사용자 갱신 시 사용하는 메소드이다.

resetRv() : 사용자가 viewtype 을 변경하거나 userList 갱신 시, 동적으로 그린 뷰를 제거하고 다시 그리기 위해 사용하는 메소드이다. mainRecyclerView 에 있는 하위 뷰를 mainRecyclerView.removeAllViewsInLayout() 메소드로 지우고 다시 생성한다. 만약 헤더 확장애니메이션이 실행되어 있는 상태인 경우, 애니메이션을 취소하고 높이를 기존 상태로 되돌린다.

startExpandAnimation() : nameRecyclerView 의 터치이벤트 발생 시, 해당 포지션의 헤더 크기를 늘리는 애니메이션을 실행한다. 단, nameRecyclerView 이외에 mainRecyclerView 도 함께 이벤트를 실행해야 하기 때문에 TouchEvent 가 발생한 포지션의 값으로 findViewHolderForAdapterPosition 메소드를 이용하여 포지션에 맞는 뷰를 불러온 뒤, 각 뷰의 높이를 300DP 로 늘리는 애니메이션을 실행하는 메소드이다.

cancelExpandAnimation() : startExpandAnimation 과 반대로 현재 늘어나 있는 포지션의 뷰를 다시 원래 크기로 되돌리는 애니메이션을 실행하는 메소드이다.

3.1.2 mainRecyclerView

mainrv 패키지는 메인 리사이클러뷰 관련 클래스가 모여있는 패키지이다.

- **MainAdapter** : mainRecyclerView 의 어댑터이며 DiffUtil 을 이용한 ListAdapter 를 사용한다. 패키지 내에 UserDiffCallback 으로 아이템의 동일 여부를 비교하며 유저 고유의 Id 와 name, scheduleList 가 모두 일치하는 경우에만 같은 아이템으로 판단한다.
- 또한 실제 사용자가 저장한 각 User 의 position 은 1 부터 시작하며 0 번째 포지션은 시간 헤더로 고정되어 있다. 따라서 2 가지타입의 뷰홀더를 별도로 제작하여 포지션별로 다른 뷰홀더를 바인딩하였다. getItemViewType 메소드를 오버라이딩 하여 position 이 0 인 경우에만 뷰타입을 0 으로 지정한 상태이다.
- **StickyHeaderItemDecoration :** ItemDecoration 을 상속하여 화면 위에 0 번째 포지션의 시간 헤더를 다시 그려주는 동작을 위한 ItemDecoration 이다. 이것으로 상단 헤더가 고정되는 효과를 구현하였다.
- **ItemContent :** content 패키지 내에는 mainRecyclerView 에서 사용자의 일정을 표기할 ContentViewHolder 와 뷰로 사용할 ItemContent 가 있다. 뷰로 사용할 ItemContent 는 기본적으로 LinearLayout 이며 뷰타입에 따라 내부 구조가 달라진다. 일정을 겹쳐서 표기하는것을 고려해야하는 일간 뷰의 경우 내부에 FrameLayout 이 있으며 주간뷰의

경우는 각 날짜에 맞춰 Vertical LinearLayout 이 추가되어 있다. 총 1 주일의 기간을 보여주어야 하므로 7 개의 LinearLayout 이 있다.

- **ContentViewHolder** 에서 사용자의 일정과 현재 날짜에 맞춰 동적으로 일정을 그린다. 일정은 TextView 로 구성되어 있다. 사용자의 일정을 그리기 위한 알고리즘은 뷰 타입에 따라 다르게 설계하였다.
- **ItemHeader :** header 패키지 내에는 상단 시간헤더를 표기할 HeaderViewHolder 와 뷰로 사용할 ItemHeader 가 있다. ItemHeader 는 LinearLayout 이며 뷰 타입에 따라 시간 텍스트를 추가하거나 날짜 텍스트를 추가한다. 주간 뷰의 경우 calendar 를 통해 요일을 판별하여 토요일인 경우 textColor 를 blue 로 지정, 일요일인 경우 textColor 를 red 로 지정하였으며 일간 뷰의 경우 0~23 시 까지의 뷰가 추가된다.

일간 뷰에서 일정을 그리는 방법

- 먼저 사용자의 스케줄목록을 순회하며 해당 시간에 맞는 뷰를 그리게 된다. 이 때, 뷰는 ItemContent 에 미리 생성해 놓은 FrameLayout 위에 TextView 를 그리게 된다. 일정의 Case 를 총 4 가지 경우로 분류한다.

1. 일정의 시작날짜, 종료날짜가 모두 현재 날짜를 벗어나지 않는 경우
   TextView 의 margin, width 를 시작시간 및 종료시간에 맞춰 설정한다. 먼저, 시작시간과 종료시간을 비교하여 사용자가 설정한 per_topheaderwidth 와 곱한 값으로 width 를 지정한다. 시작시간 * per_topheaderwidth 로 marginstart 를 지정한다. 이렇게하여 하루 이내의 일정에 대한 처리를 구현하였다.
2. 일정의 시작날짜가 현재날짜이고 종료날짜가 현재날짜보다 이후인 경우
   TextView 의 margin 은 1 번 케이스와 마찬가지로 시작시간에 맞춰 설정한다. 하지만 종료시간을 따로 비교할 필요 없이 width 를 MATCH_PARENT 로 지정한다.
3. 오늘 날짜가 일정의 시작 날짜 이후이고 종료날짜 이전인 경우
   즉 오늘 날짜에선 해당 일정이 종일 일정인 경우이다. 이 경우에는 더 간단하게 margin 없이 width 를 MATCH_PARENT 로 지정한다.
4. 오늘 날짜가 일정의 시작 날짜 이후이고 종료날짜가 오늘 날짜인 경우
   이 경우에는 width 를 종료시간 * per_topheaderwidth 로 지정하기만 하면 된다. 결과적으로 0 시~종료시간 까지가 해당 일정의 영역으로 지정되며 사용자에게 표기된다.

일간 뷰에서 겹치는 일정에 대한 처리

겹치는 일정은 Case에 관계없이 처리해야 하기때문에 모든 케이스에 공통적으로 적용된다. 알고리즘은 직접 구상하여 구현하였으며 아래와 같이 처리한다. 먼저 시간순으로 정렬된 일정에서 현

재 일정의 이전 일정중 겹치는 일정의 개수를 저장한다. (cnt_prioroverlap). 이후 일정중 겹치는 일정의 개수를 저장한다(cnt_lateroverlap)
해당 일정의 높이는 전체 높이에서 (cnt_prioroverlap+cnt_lateroverlap+1)을 나눈값이며 margin을 높이*cnt_prioroverlapd으로 지정하였다. 하지만 테스트 결과, 겹치는 일정의 수가 매우 많은 경우 일정의 height가 너무 작아지는 문제가 있어 일정이 아무리 많이 겹치더라도 기존 높이의 1/3미만으로 떨어지지는 않도록 하였다. 그 결과 아래 그림과 같이 처리된다.

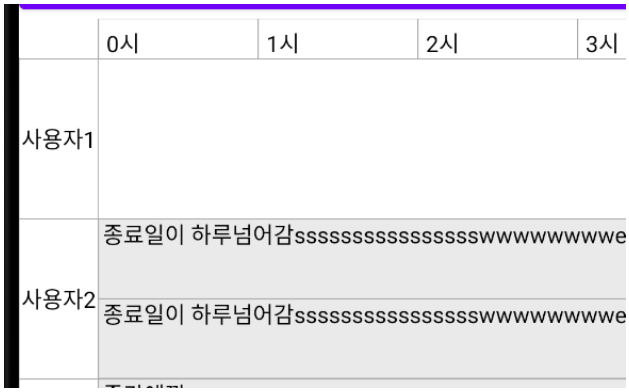

그림 5

일정이 매우 많이 겹치는 경우에도 아래와 같이 일정을 확인할 수 있다.

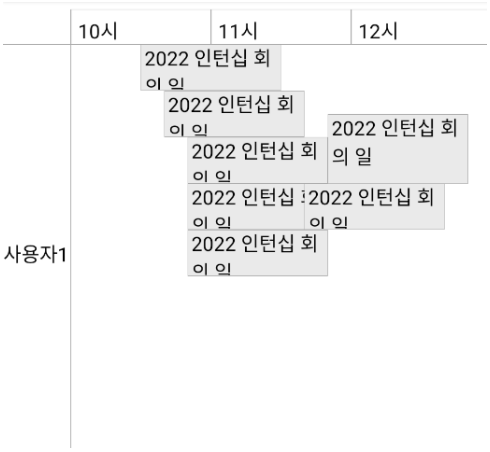

그림 6

주간 뷰에서 일정을 그리는 방법

주간뷰는 기존 ItemContent 에서 기존 뷰 내에 7 개의 LinearLayout 을 추가하고 이를 List 로 구성해놓았다. 이 레이아웃 리스트의 이름은 week_layoutList 이며 주간 뷰에서 현재 날짜와 일정의 시작, 종료날짜를 적절히 비교하여 알맞은 Index 의 week_layoutList 에 일정을 쌓아나가는 방법으로 구현하였다. 주간 뷰도 총 4 개의 Case 로 구성되어 있다. 단 주간뷰의 경우 케이스별로

해당 날짜의 일정이 종일인 경우가 있고, 해당 날짜가 일정의 종료일만을 포함하거나 시작일만을 포함하는 경우가 있다. 현재 PC 에서는 이러한 경우를 아래와 같이 처리한다.

그림 7

현재 일정은 1 월 31 일 22:00~ 2 월 2 일 10:00 까지의 일정이다. 시작일만을 포함하는 일정의 경우 해당 일정의 시작시간만을 표기하고, 사이에 끼어 종일에 해당하는 일정은 시간 대신 "종일"이라는 텍스트를 표기하고, 종료일에는 종료시간만을 표기하도록 되어있다. 이에맞춰 라이브러리에도 같은 방법을 적용하기 위한 알고리즘을 구상하고 구현하였다.

1. 일정이 하루 이내의 일정이며 그 시간이 현재 날짜로부터 1 주일 이내로 있는 경우

2. 일정이 하루를 넘어가며 시작, 종료 시간이 모두 1 주일 내에 있는 경우

3. 일정이 하루를 넘어가며 시작 시간은 1 주일 내에 있지만 종료시간이 1 주일을 벗어나는 경우

4. 일정이 하루를 넘어가며 시작 시간이 현재날짜 이전이며 종료시간이 1 주일 내에 있는 경우

5. 일정의 시작, 종료시간의 차이가 8 일 이상이며 현재 날짜가 일정 사이에 있는 경우

먼저, 기본적인 뷰를 그리는 방법은 모두 동일하다. 현재 날짜와 해당 날짜의 차이를 Difference 라고 한다면 미리 정의해놓은 week_layoutList 의 Difference 인덱스에 텍스트뷰를 추가하는 것이다. 단, 케이스에 따라 텍스트를 시작 시간만 표기하거나, 종일로 표기하거나, 종료시간만 표기하는 것이다. 하루 이상의 일정의 경우, for 문을 통해 해당하는 일정에 맞춰 모두 일정 텍스트뷰를 추가한다. 예를들어, 그림 7 에서 보았던 3 일에 걸친 일정은 현재 구현한 뷰에서 그림 8 과 같이 구현된다.

그림 8

일정 클릭 리스너

일정 클릭 리스너는 기본적으로 mainRecyclerView 에서 mainAdapter 의 파라미터로 ViewHolder 까지 전달된다. 기본적으로 ContentViewHolder 는 View.OnClickListener 를 상속받아 onClick 메소드를 오버라이딩 하여 구현한다. 일정을 클릭할 때 동적으로 그린 TextView 마다 setOnClickListener(this)를 통해 리스너를 명시해야 하므로 이를 담을 scheduleListView = ArrayList<TextView>()를 선언하고, 뷰를 동적으로 그릴때마다 리스트에 추가한다.

```
override fun onClick(v: View) {
    if(v in scheduleListView){
        onClickScheduleListener.onClickSchedule( position: adapterPosition-1,map.getOrDefault(scheduleListView.indexOf(v), defaultValue: -1))
    }
    else{
        onClickScheduleListener.onClickEmptySchedule()
    }

}
```

그림 9

만약 클릭 이벤트가 발생한 뷰가 scheduleListView 에 있다면, onClickScheduleListener 의 onClickSchedule 메소드를 실행한다. 이때, adapterPosition 과 해당 유저의 schedule 에서 클릭한 schedule 의 index 를 전달한다. map 은 ConcurrentHashMap<Int, Int>로 구성되어 있으며 현재 추가한 scheduleListView 에서의 Index 로 유저의 scheduleList 의 Index 를 찾기 위해 사용된다.

이 값을 전달받고 스케줄의 id 를 조회하고자 한다면 userList[position].scheduleList[scheduleidx]로 해당 스케줄에 접근할 수 있다.

클릭 이벤트가 발생한 뷰가 scheduleListView 에 포함되지 않았다면, 빈 스케줄을 클릭한 이벤트이므로 onClickScheduleListener.onClickEmptySchedule 메소드를 실행한다.

3.1.3 nameRecyclerView(좌측 헤더)

leftrv 패키지 내에 좌측 헤더 관련 리사이클러뷰의 클래스들을 정리하였다. nameRecyclerView 는 기존 userList 에서 filter{it.name}을 이용하여 이름만을 리스트로 추출하여 사용한다.

- **LeftAdapter** : nameRecyclerView 의 어댑터이며 DiffUtil 을 이용한 ListAdapter 를 사용한다. 패키지 내에 NameDiffCallback 으로 아이템의 동일 여부를 비교하며 유저의 name 만 비교하여 같은 아이템인지 여부를 판단한다.

- **ItemName** : name 패키지 내에는 이름을 표기할 NameViewHolder 와 뷰로 사용할 ItemName 이 있다. ItemName 은 LinearLayout 이며 이름을 표기할 TextView 를 하위 뷰로 가지고 있다.

3.1.4 model

라이브러리에서 사용하는 데이터 모델은 크게 3 가지 이다. 시간을 표기하기위한 Time 객체, 스케줄의 정보를 담은 Schedule 객체, 사용자의 정보를 담은 User 객체가 이에 해당한다.

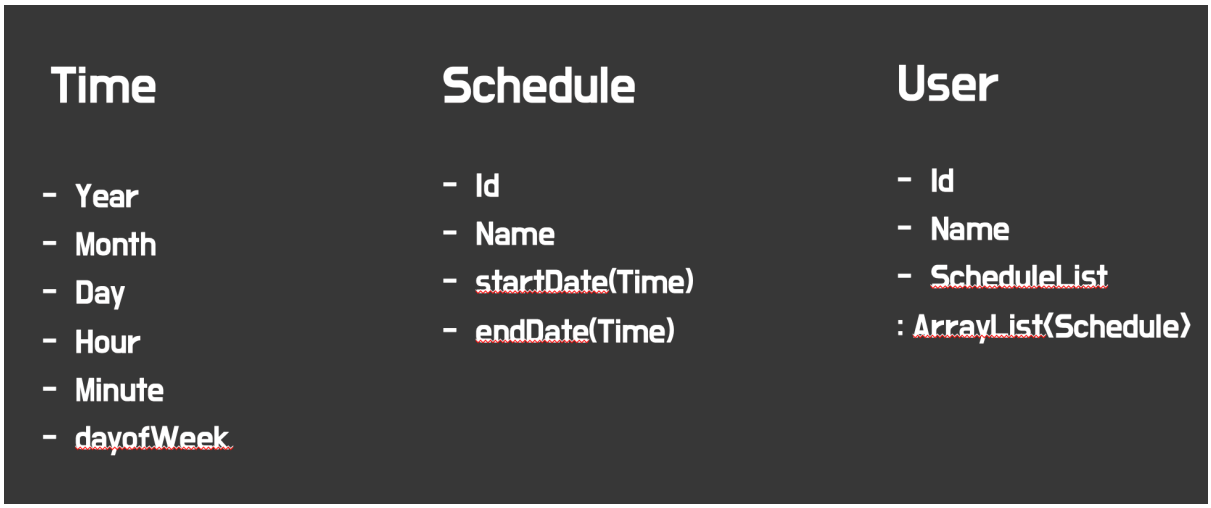

그림 10

4. Class Diagram

4.1 Library Structure

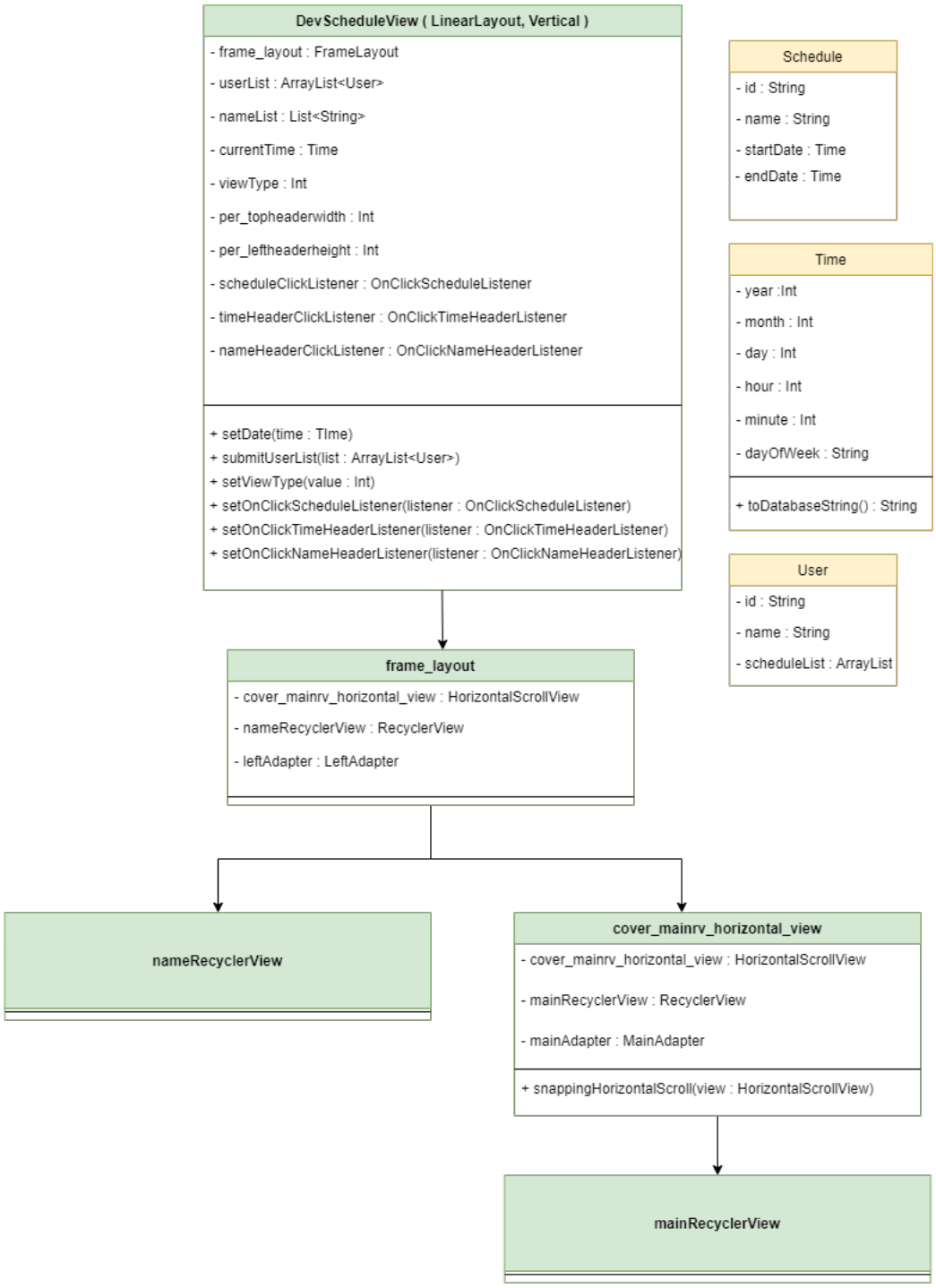

그림 11

4.2 nameRecyclerView

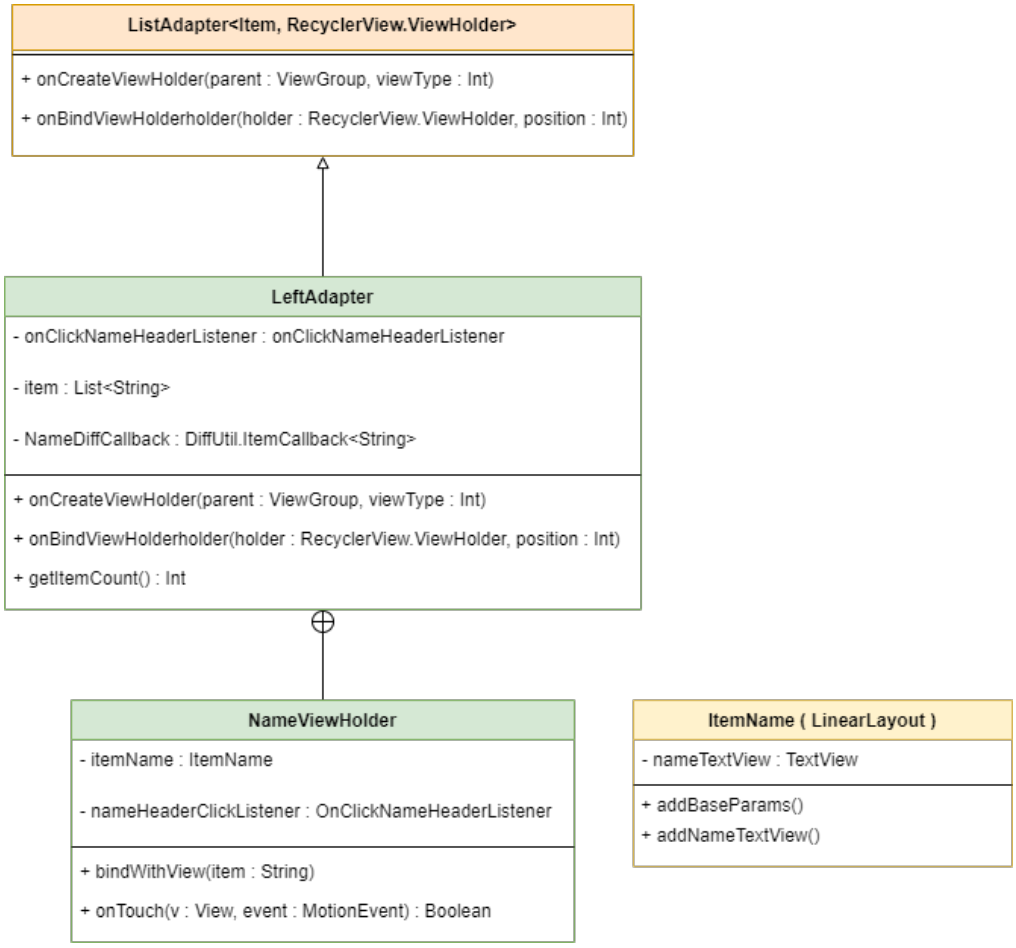

그림 12

4.3 mainRecyclerView

그림 13