ListView & RecycleView

- Adapter
  - To use ListView, we need extends/override BaseAdapter
  - To use RecycleView, we need extends/override RecyclerView.Adapter and RecyclerView.ViewHolder
- ViewHolder(ViewModel)
  - ListView has customize ViewHolder and convertView, do null check, if-else
  - RecycleView ViewHolder is pre-defined
- Layout
  - RecyclerView use LayoutManager for multiple layout appearance(Vertical, Horizontal, Grid*2: uniform/Staggered)
  - ListView has vertical layout only
- Item reuse
  - RecyclerView handled item reuse
  - ListView need manually `setTag()` and `getTag()`
- EmptyView
  - ListView has `setEmptyView()`
  - RecycleView need manually check null data
- Header and Footer
  - ListView has `addHeaderView()` and `addFooterView()`
  - RecycleView use ViewHolder's Type and View for Header and Footter implementation
- Item refresh
  - ListView has `notifyDataSetChanged()` for full refresh
  - RecycleView has `notifyItemChanged()`/`notifyDataInserted()`/`notifyItemMoved()` for partial refresh
- Item animation
  - RecycleView API: `setItemAnimator()` + `RecyclerView.ItemAnimator`
  - ListView implementat item animation in Adapter
- Item Click
  - ListView has `onItemClickListener`, `onItemLongClickListener`, `onItemSelectedListener`; but normally use `getItemId()` for identifier instead of position since HeaderView and FooterView take positions

    - RecyclerView API：`addOnItemTouchListener()` + Gesture Detector; or use customized item click callback

- Item Touch
    - RecycleView has ItemTouchHelper for `onMove()`, `onSwiped()`: `ItemTouchHelper.SimpleCallback()` + `attachToRecyclerView()`

- Nested Scrolling
    - RecyclerView has `NestedScrollingChild` and `NestedScrollingParent`
    - ListView framework doesn't support