Android Apps with Kotlin: RecyclerView and Navigation Drawer

WORKING WITH RECYCLERVIEW AND CARDVIEW

Jim Wilson
MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim blog.jwhh.com

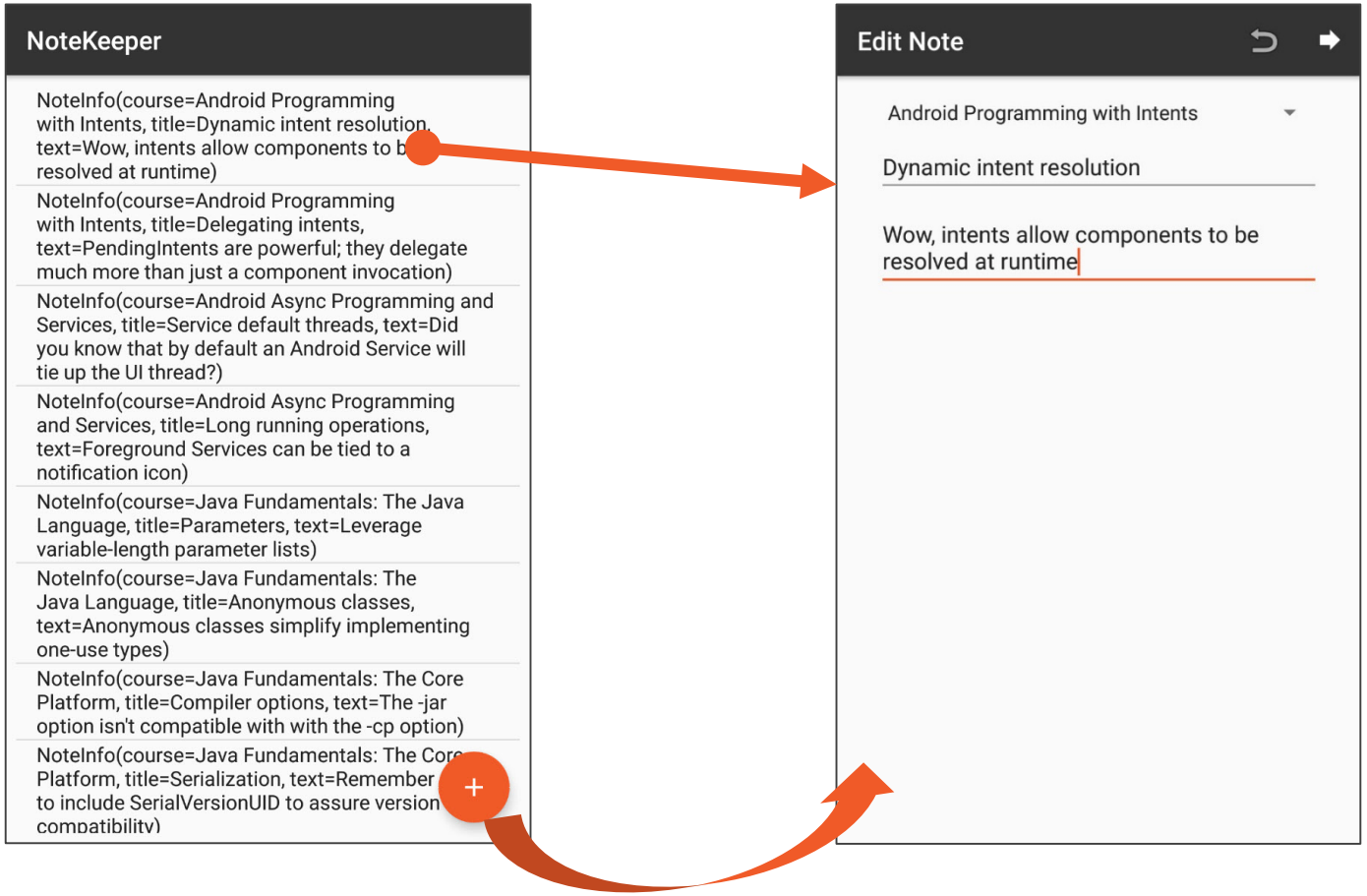
What to Expect from This Course

- Using RecyclerView and CardView
- Binding data with RecyclerView.Adapter
- Understanding Drawer Navigation
- Implementing Drawer Navigation
- Testing RecyclerView and Drawer Navigation

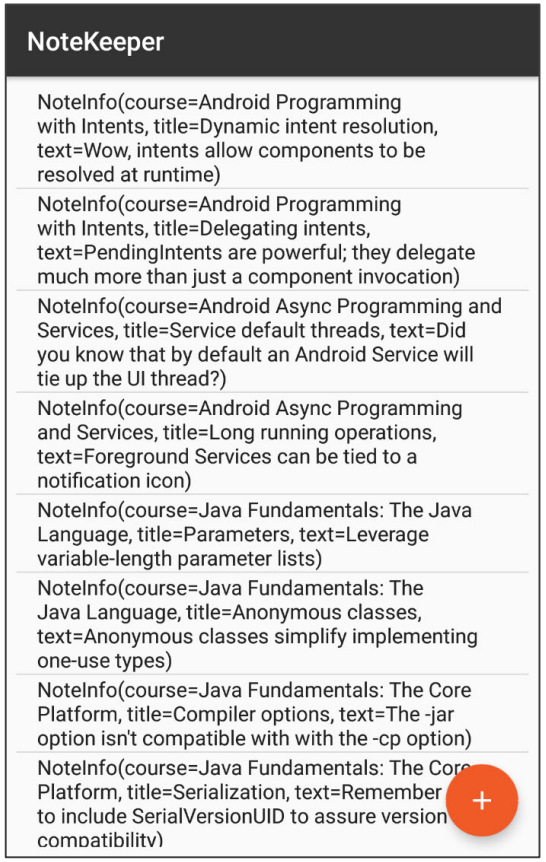
What to Expect from This Module

- Overview of our demo app
- RecyclerView components
- Card-like item appearance with CardView
- Add a RecyclerView to our demo app
- Item positioning with LayoutManager

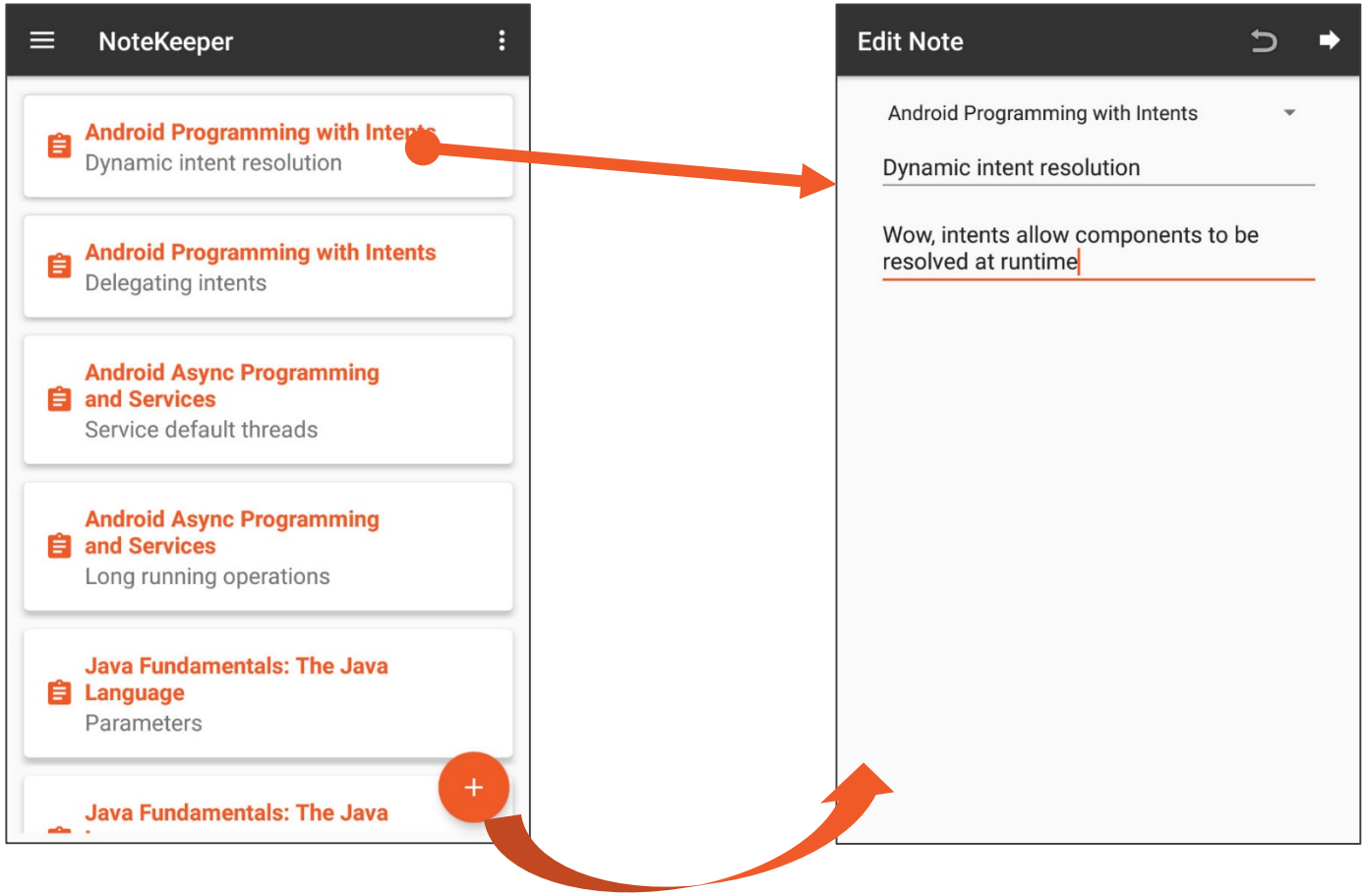
Overview of Our App

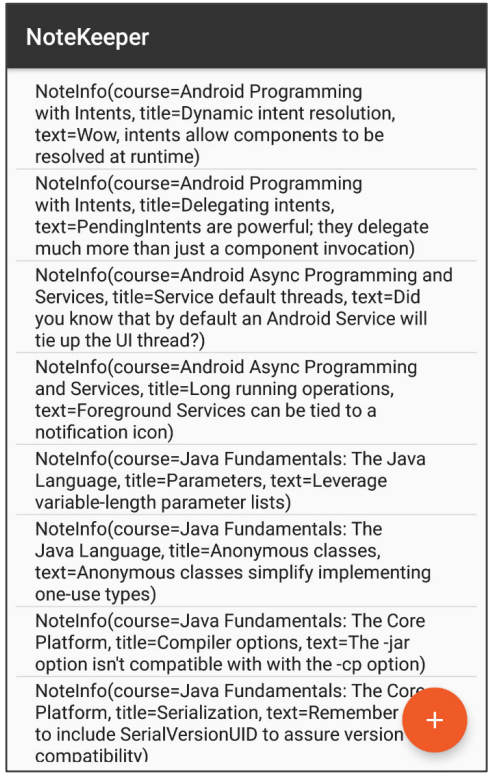


Overview of Our App



Overview of Our App





Displaying lists of data very common

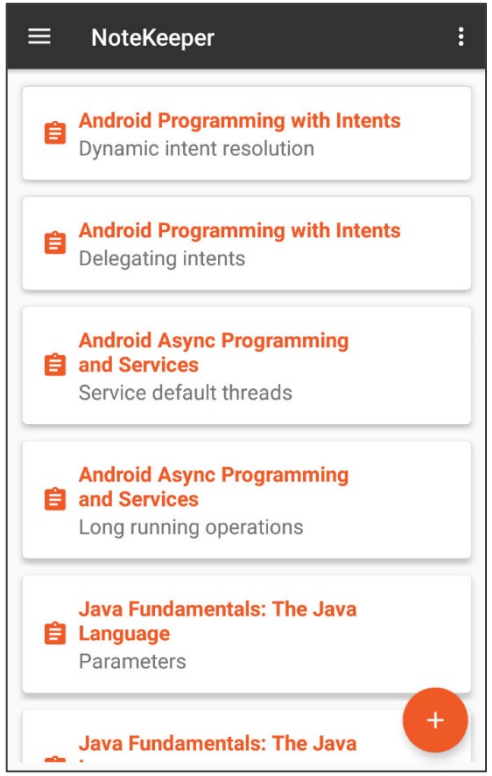
- Historically relied on ListView

ListView has limitations

- Always displays as vertical list
- Can be challenging to customize
- Performance challenges in some cases

ListView and modern app expectations

- Need a solution with more flexibility



RecyclerView is designed for modern apps

- Extremely flexible

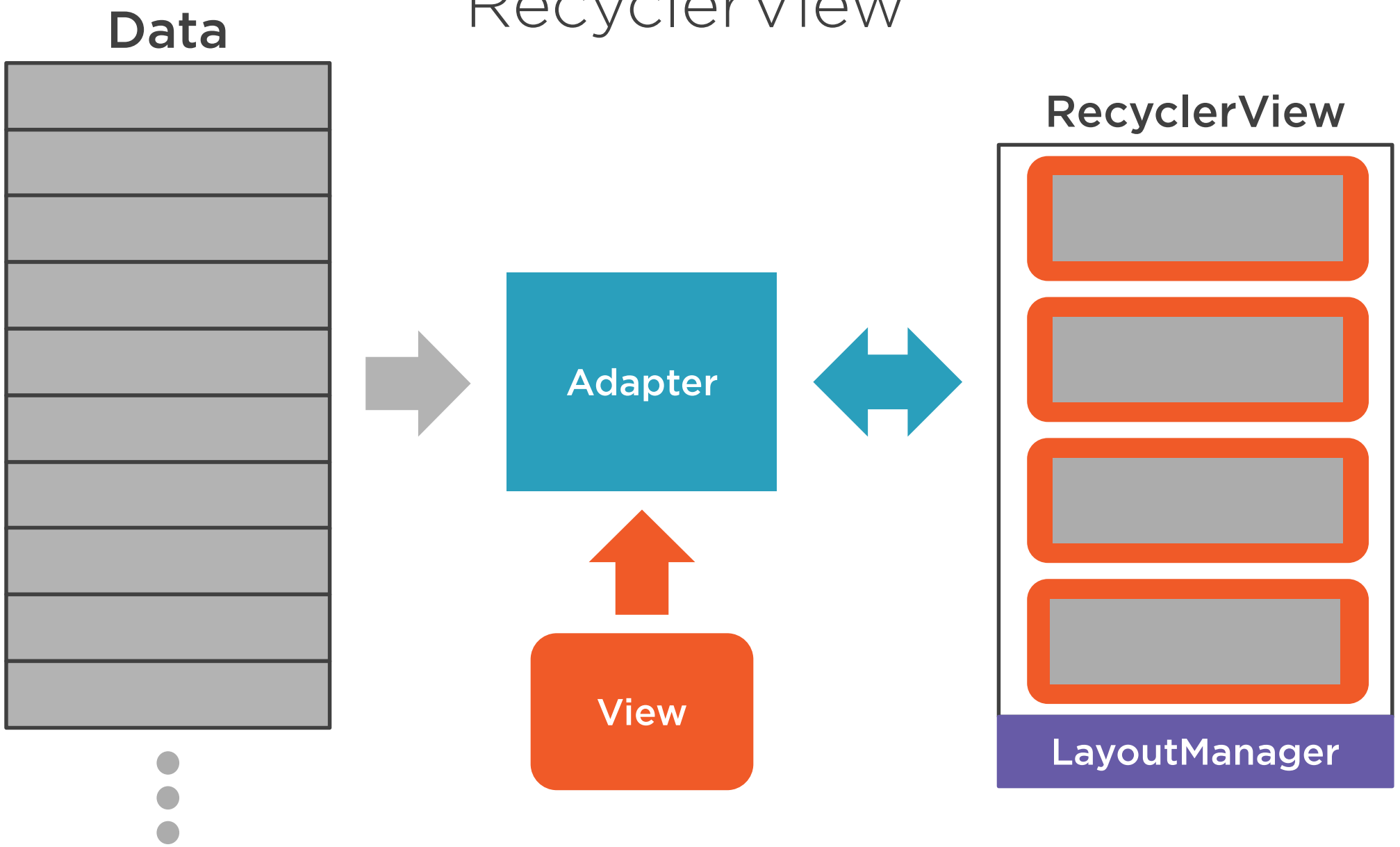
List display divided into distinct phases

- Each phase offers chance to customize

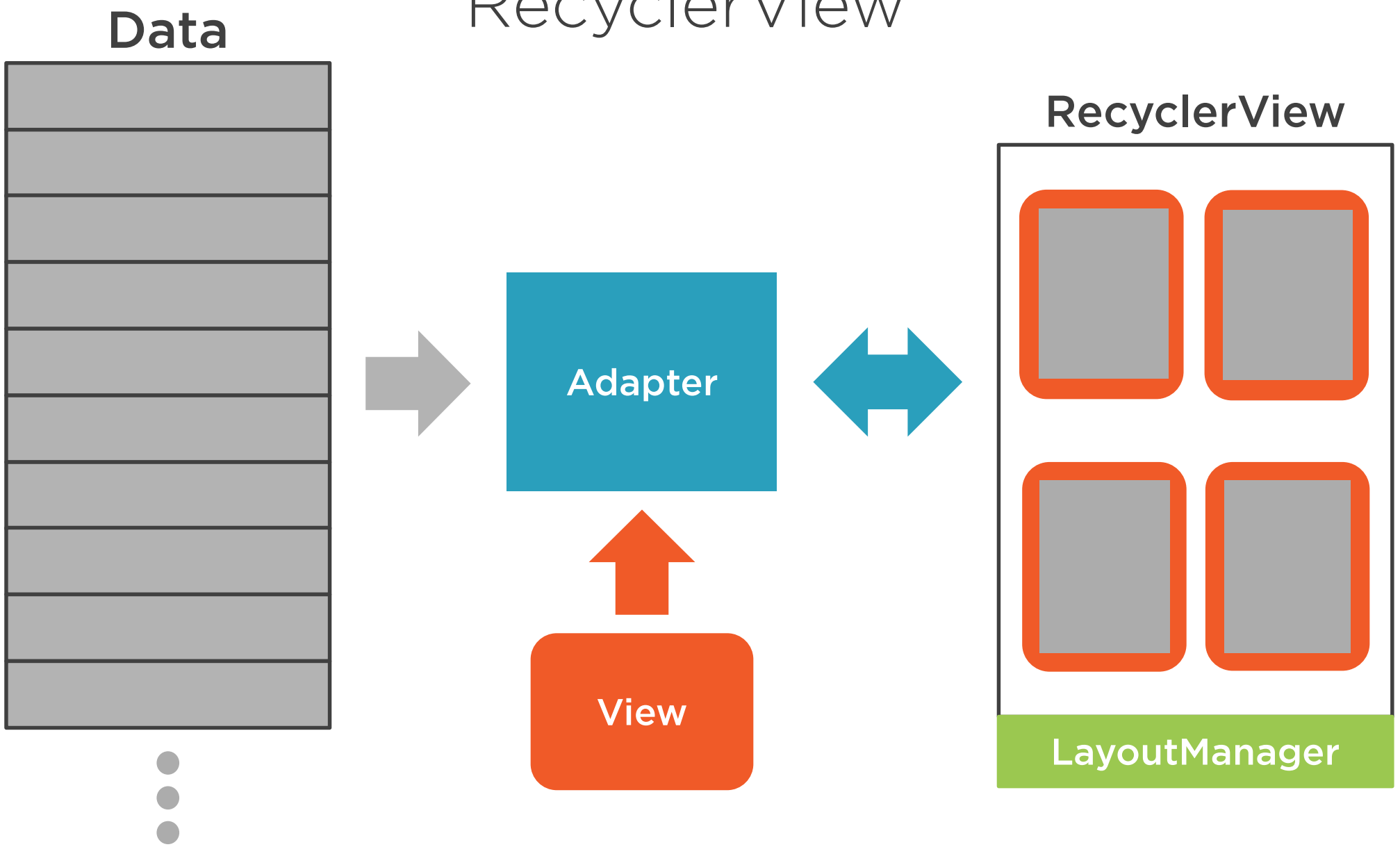
Provides efficient display management

- Separates details of data from display

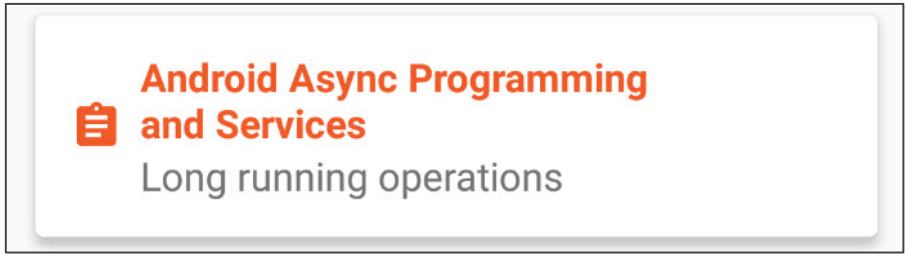
RecyclerView



RecyclerView



Developing RecyclerView Components



Design the RecyclerView

Handled much like any other view

Usually part of a layout resource

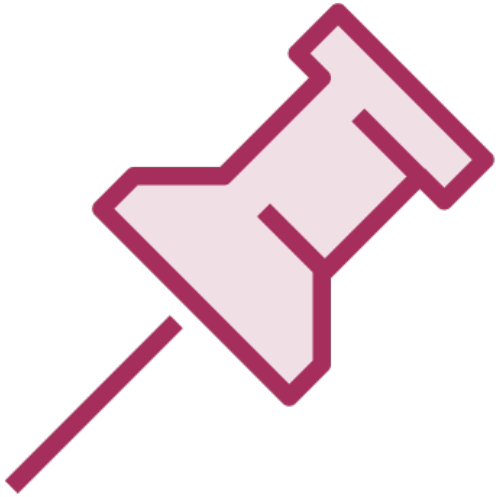
Design the item view

Controls appearance of individual item

Usually a layout resource

Separate resource from RecyclerView

Developing RecyclerView Components



Create and associate layout manager

Controls item arrangement and positioning

Create and associate adapter

Constructs item view instances

Manages data interaction

Associates data items with item views

Layout Manager

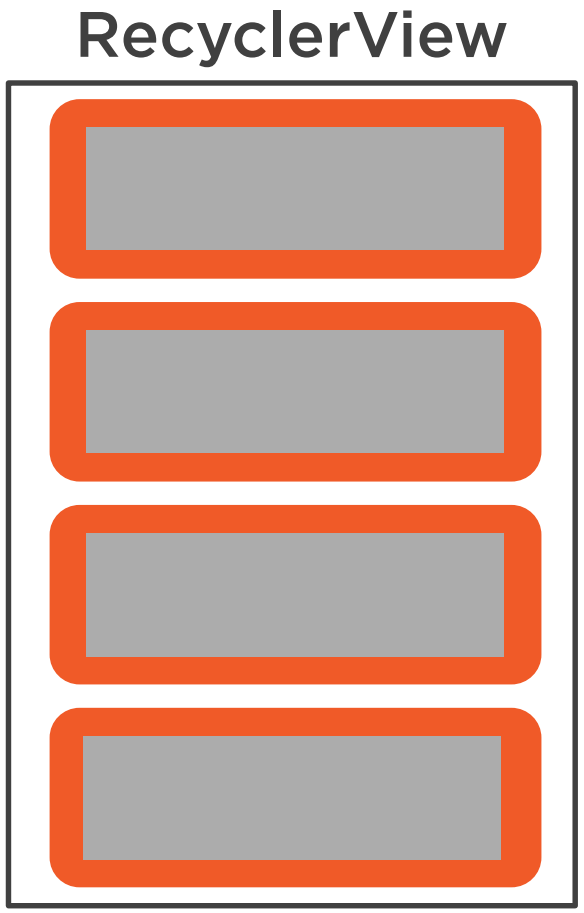
RecyclerView.LayoutManager

- Base class for layout managers
- Extend to create custom layout manager

Android provides several implementations

- Handle most common scenarios
- Support vertical & horizontal orientation

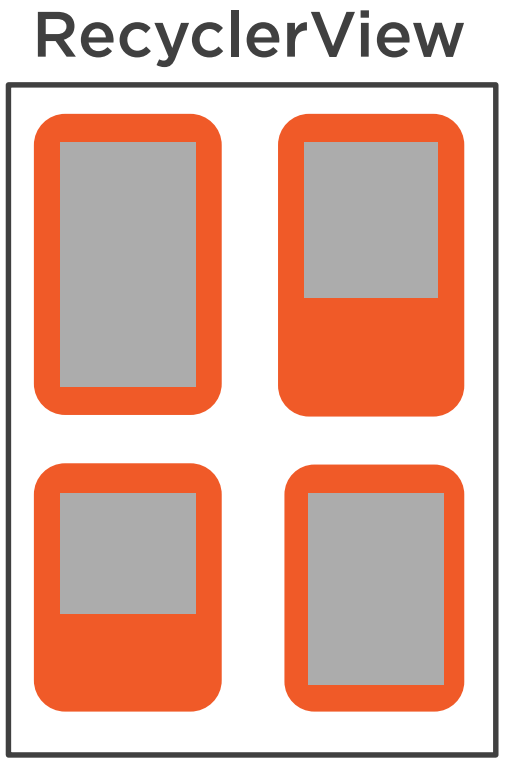
LinearLayoutManager



Items organized as linear list

- Similar to ListView

GridLayoutManager



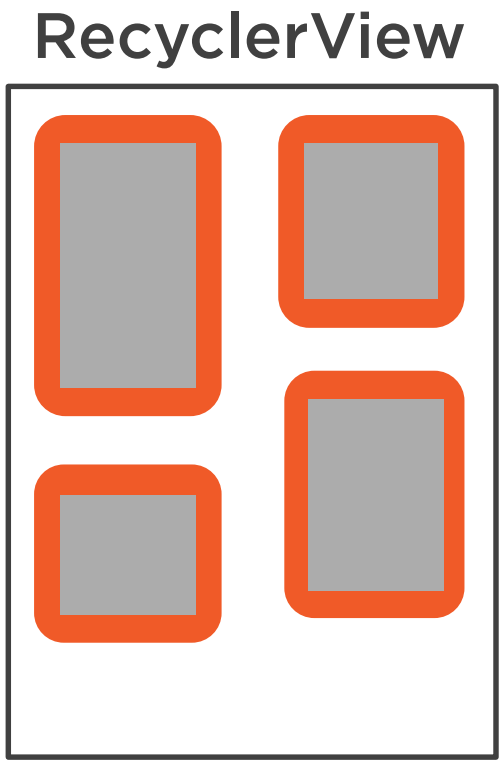
Items organized as a grid

- Adjacent items consistently sized

Can specify span

- Columns for vertical orientation
- Rows for horizontal orientation

StaggeredGridLayoutManager



Items organized as a grid

- Each item individually sized
- Can specify span

Summary

RecyclerView benefits

- Extremely flexible
- Divides display into phases
- Each phase can be customized
- Efficient display management

Summary

RecyclerView

- Presents list of data

LayoutManager

- Handles positioning of items

Adapter

- Creates item views
- Associates data with item views

Summary

Design the RecyclerView

- Normally part of layout resource

Design item view

- Normally a layout resource
- Separate resource from RecyclerView

Create and associate layout manager

- Normally use one of the built-in ones

Create and associate the adapter

- Normally need to build this ourselves