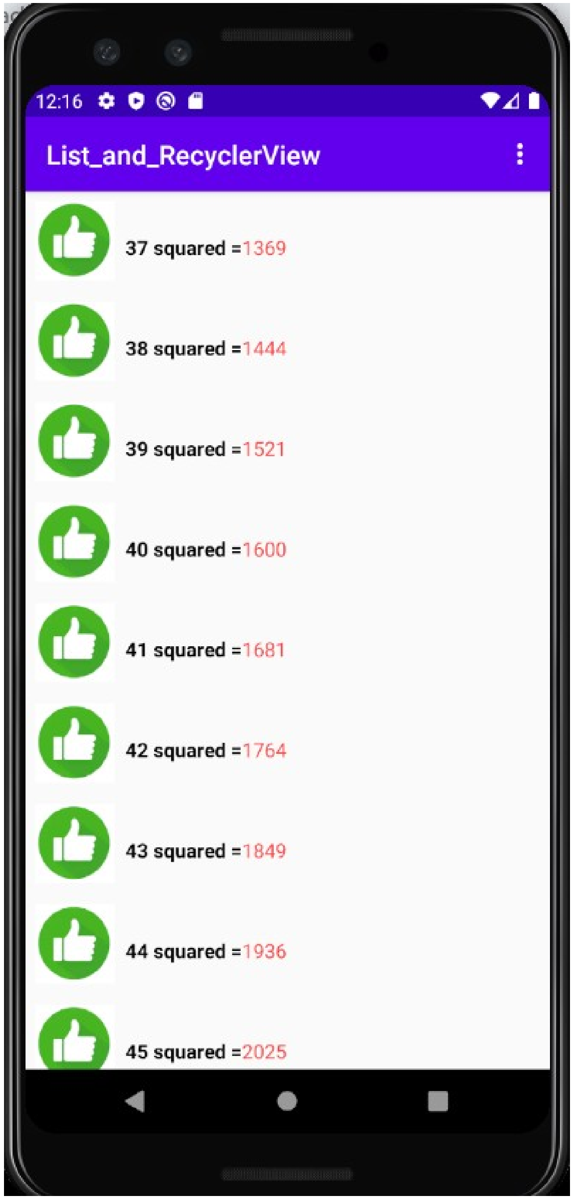
RecyclerView

Topics

- RecyclerView
- Adapters
- Lab
- Sorting
- Listeners (see offline content)

RecyclerView

- Common data pattern
- Scrolling list of data
- MVC design pattern
  - Model – the data
  - View – UI
  - Controller – Logic



37 squared =1369

38 squared =1444

39 squared =1521

40 squared =1600

41 squared =1681

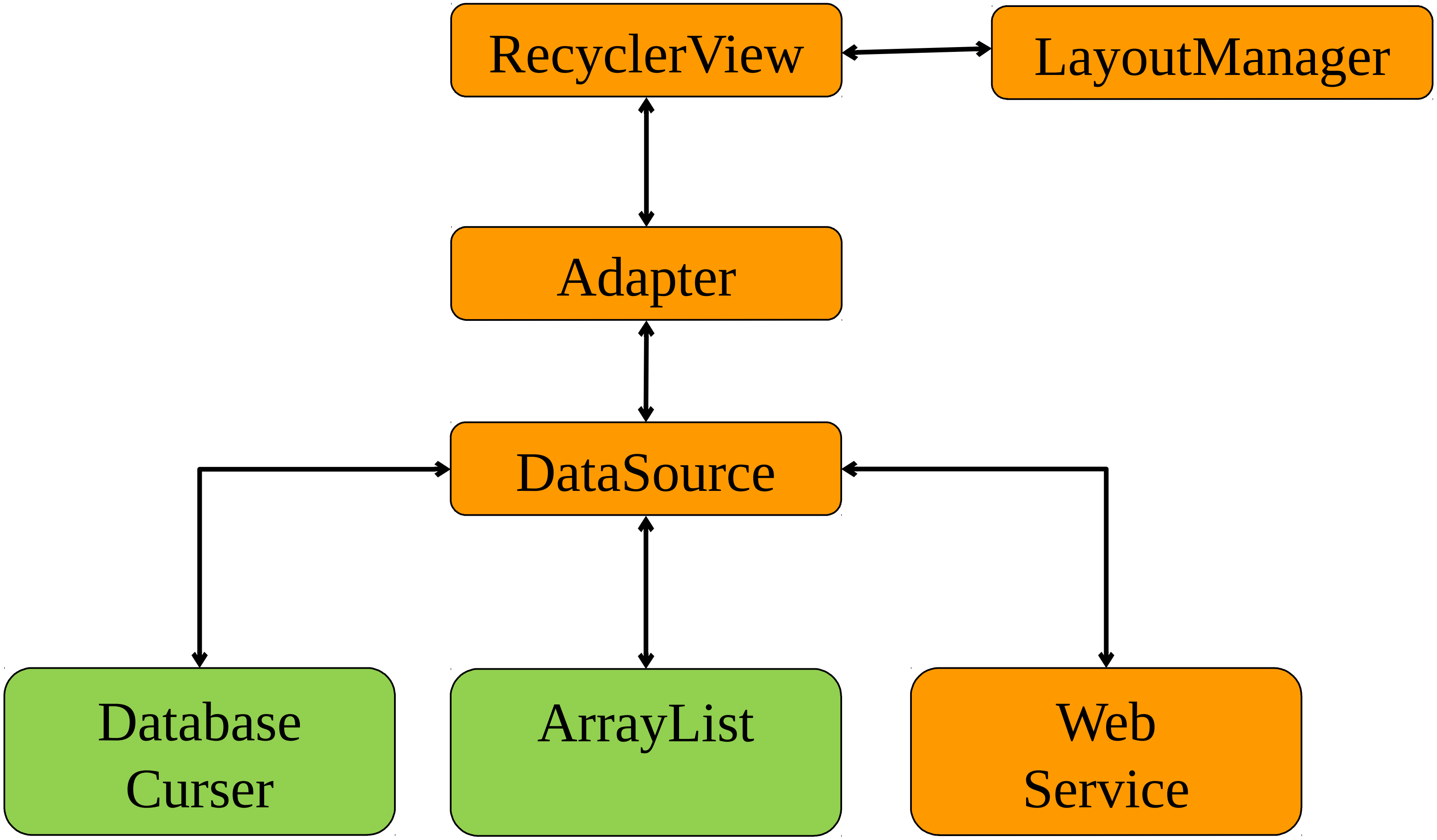
42 squared =1764

43 squared =1849

44 squared =1936

45 squared =2025

RecyclerView Overview



RecyclerView

LayoutManager

Adapter

DataSource

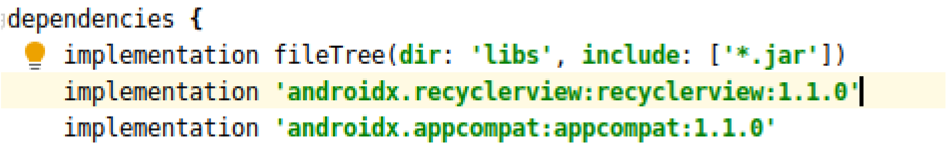
Database Curser

ArrayList

Web Service

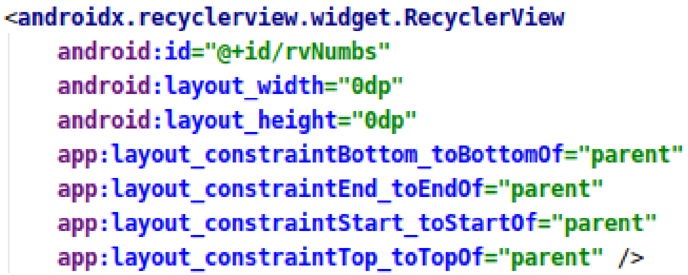
RecyclerView

- **To create;**
  - Add a dependency in your build.gradle(app)



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.recyclerview:recyclerview:1.1.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
```

  - Add a RecyclerView to your layout of interest

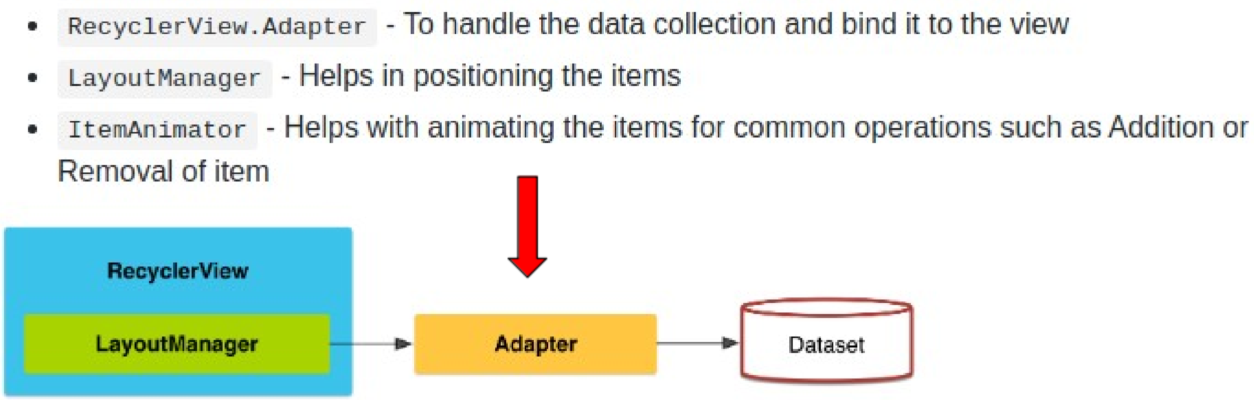


```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rvNumbs"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

- **To access data use adapters**

Adapters again

- **Manages datasource for a view**
- **Consistent access protocol**
- **Easy datasource swapping**
- **Used by a lots of things**
  - RecyclerView, ViewPager, Spinner etc..



- `RecyclerView.Adapter` - To handle the data collection and bind it to the view
- `LayoutManager` - Helps in positioning the items
- `ItemAnimator` - Helps with animating the items for common operations such as Addition or Removal of item

RecyclerView - Recipe

- **Define datasource**
- **Define what each row in the list should look like**
    - Add layout in Res\layout (row_layout.xml)
- **Define a helper class, ViewHolder that extends from RecyclerView.ViewHolder.**
    - Holds references to all views of interest for a particular ‘chunk’ of data
- **Create class that extends RecyclerView.Adapter and fill in required methods**
    - OnCreateViewHolder - creates a new ViewHolder
    - OnBindViewHolder – reuses an existing ViewHolder
    - GetItemCount - Gets the number of expected rows (or items)
    -

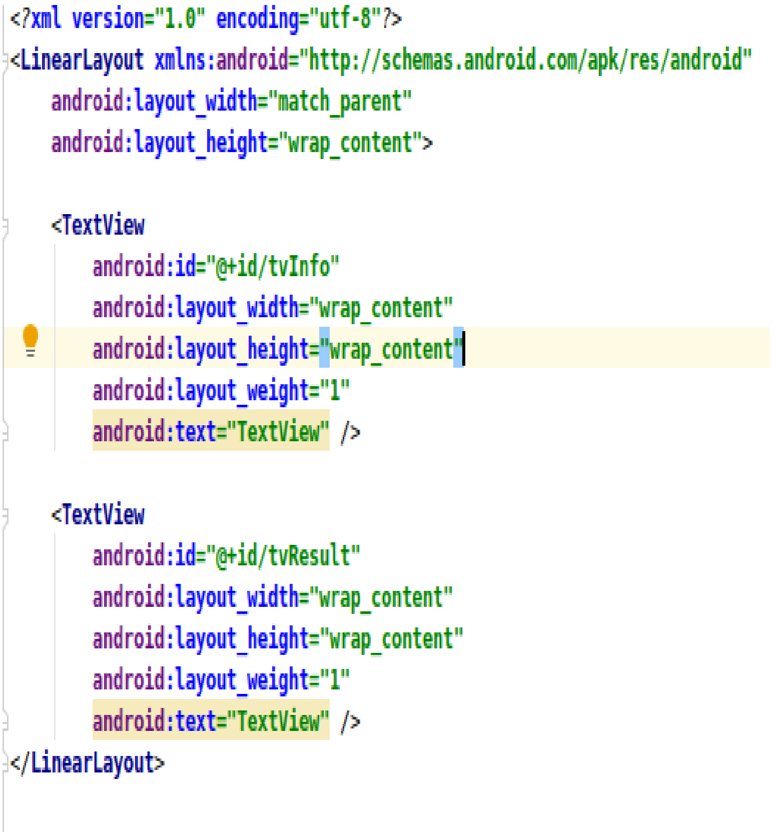
RecyclerView - Recipe

- In Activity
    - Get ref to RecyclerView
    - Create Adapter
    - Choose a layout manager (determines how data is displayed)
    - Bind Adapter to RecyclerView

RecyclerView - ViewHolder?

- Each row in the List is described by row_layout.xml.
- The Viewholder manages references to each view in that xml

row_layout.xml

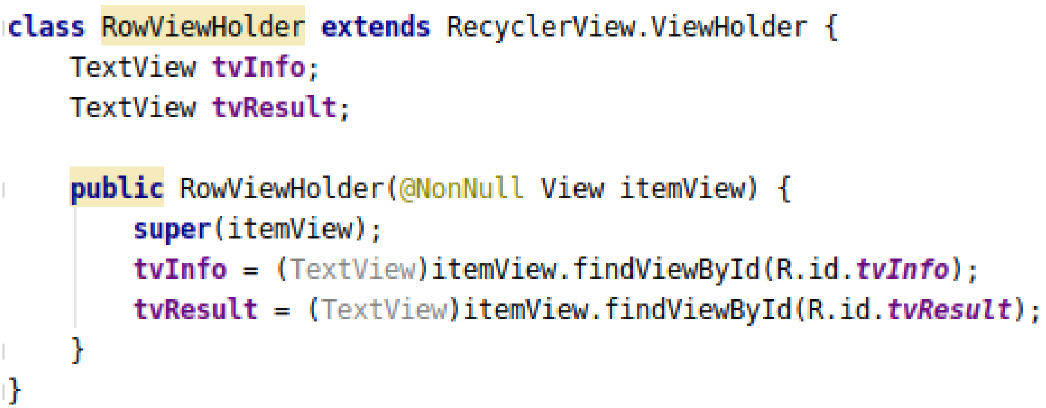


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/tvInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="TextView" />

    <TextView
        android:id="@+id/tvResult"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="TextView" />
</LinearLayout>
```

ViewHolder

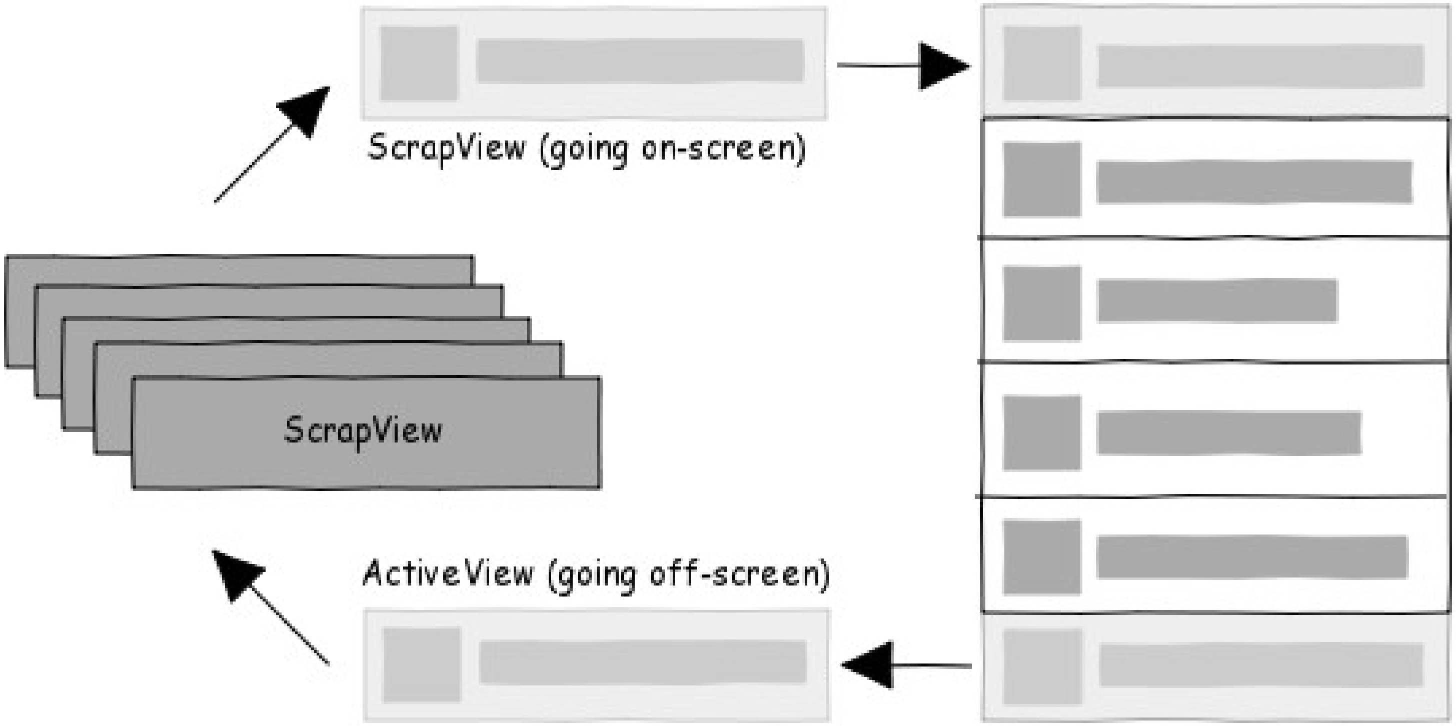


```java
class RowViewHolder extends RecyclerView.ViewHolder {
    TextView tvInfo;
    TextView tvResult;

    public RowViewHolder(@NonNull View itemView) {
        super(itemView);
        tvInfo = (TextView)itemView.findViewById(R.id.tvInfo);
        tvResult = (TextView)itemView.findViewById(R.id.tvResult);
    }
}
```

RecyclerView - ViewHolder

- *RecyclerView tries to do as few view inflations as possible because inflating row_layout.xml and getting references to its views are expensive.*
- *The ViewHolder does all this once*
- *And then is recycled and reused when the row scrolls off the screen*



Lab

- **See ‘InClass Lab: RecyclerView...” online**
- **Both the Lab and the solution**

Sorting List

- **Sort underlying datastructure**
  - How? Collection.sort(myList) Collection.reverse(myList)
- **What about noncomparable or complex objects?**
- **Use comparator interface on data**
  - Define class that implements comparator
- **Sort it when necessary**
- **Call notifyDataSetChanged() to refresh adapter after sort**

Listeners

- **Responding to List touch events**
- **See https://github.com/codepath/android_guides/wiki/Using-the-RecyclerView**
- **Section titled ‘Attaching Click Handlers to Items’**