Royal University of Bhutan

Unit II:
Fragments and Recycler View

CTE308- AS2025

Tutor: Pema Galey

#17682761

OUTLINES

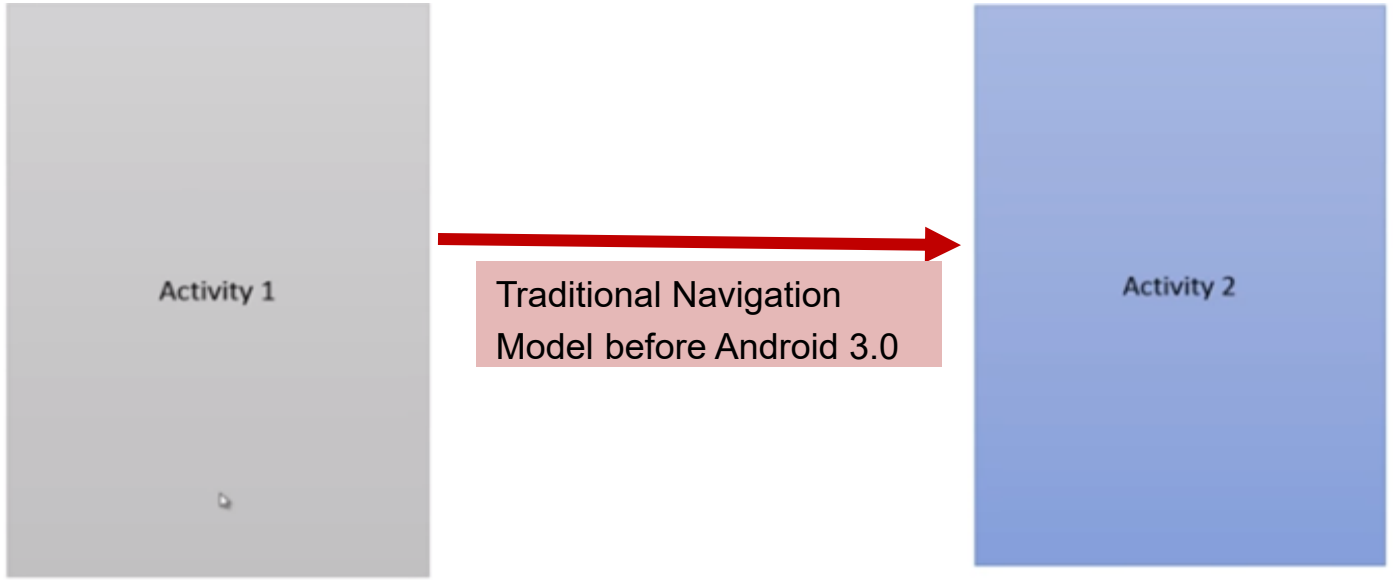
- Fragment
- Recycler View

Introduction to Fragment

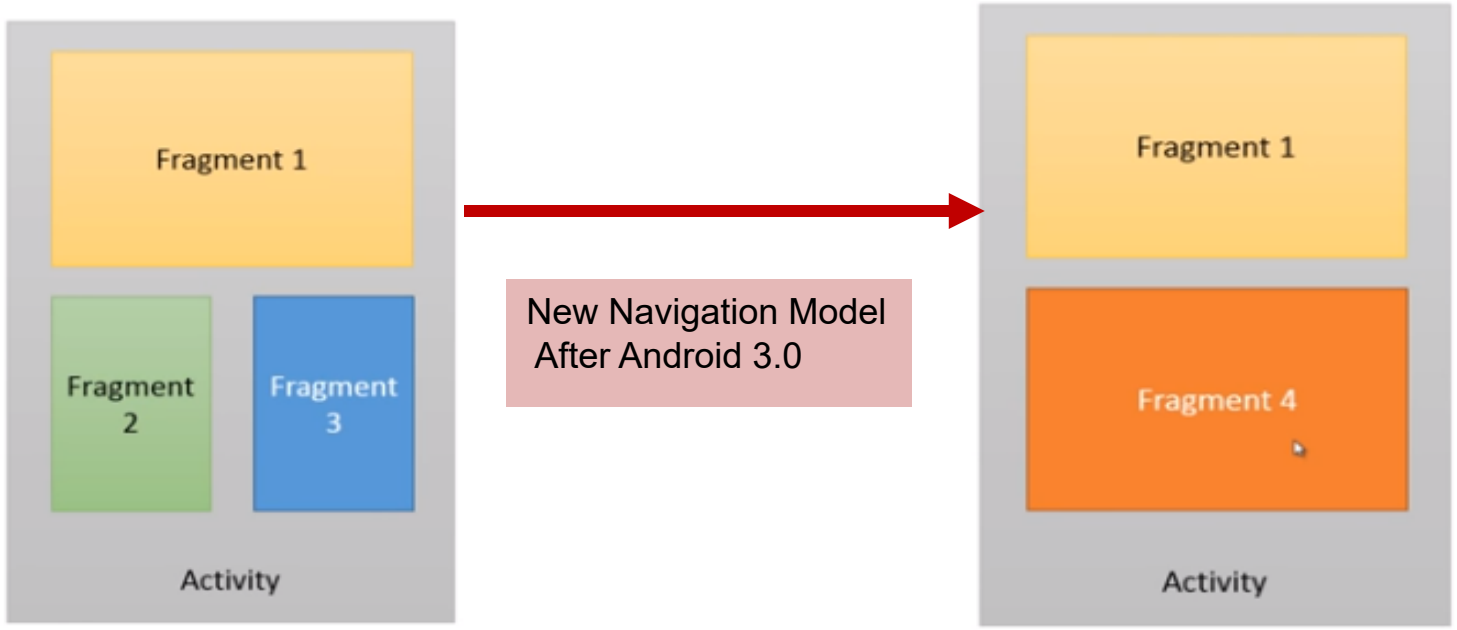
- A Fragment represents a behavior or a portion of user interface in a Fragment Activity.
- You can combine multiple fragments in a single activity to build a multi-pane UI and reuse a fragment in multiple activities.
- You can think of a fragment as a modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running.
  - It is like a "sub activity" that you can reuse in different activities.

Introduction to Fragment

- Fragment Model
  - What is Fragment?



Fragment in New Model

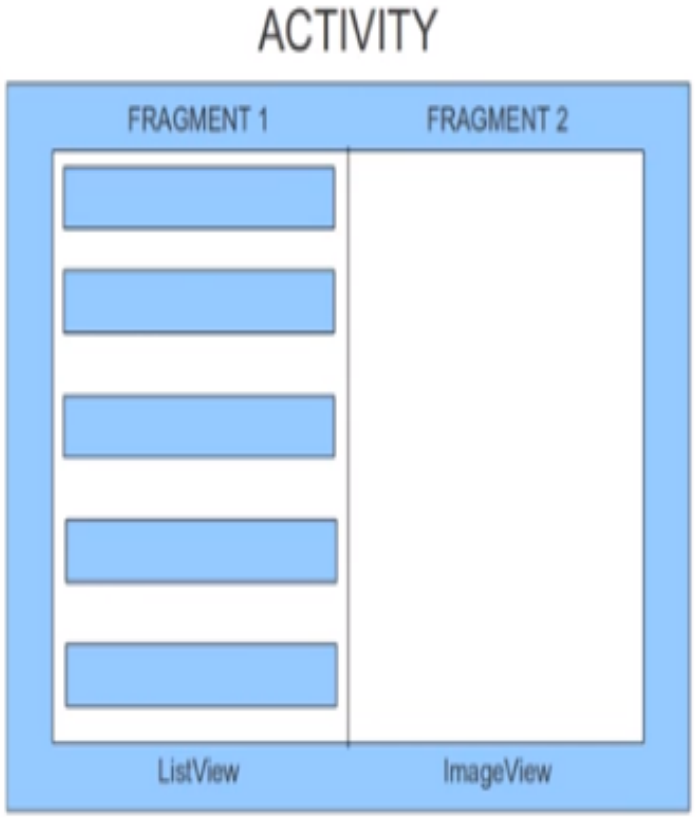


Perform Action on Fragment 1
Fragment 2 and fragment 3 disappear and fragment 4 is displayed.

Fragment

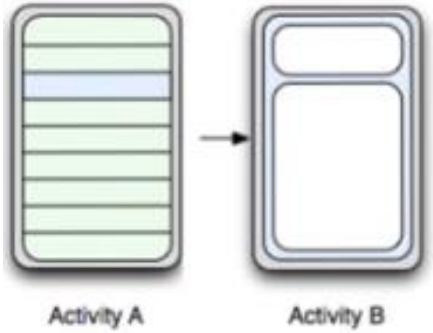
What is Fragment?

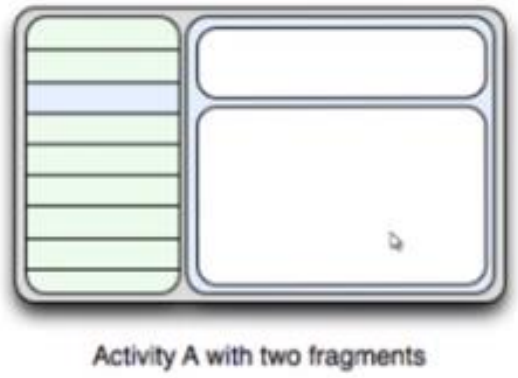
- Fragment is a small chunk of UI
- It has its own lifecycle.
- It can process its own events.
- It can be added or removed while activity run.
- It was introduced in Honeycomb API 11.
- You can use Fragments on older devices using a support Library from 1.6 to 2.3



Why Fragment?

- Why do you need Fragment?
  - To combine several fragments in a single activity
  - Reuse the same fragment across several activities
  - Make better use of larger screen space on tablets
  - Support different layout on portrait and landscape mode

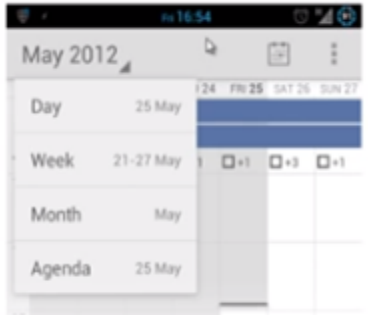


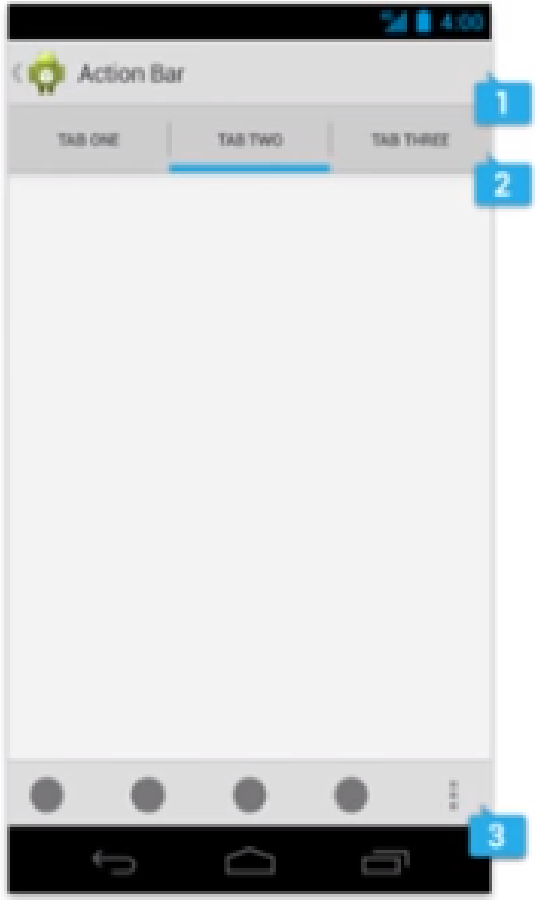


Fragment

Uses Fragment

- Flexible user interfaces across different screen sizes
- Fix/Scrolling/swap tab display
- Dialog boxes are Fragments
- Action Bar customization with the list and table modes.





Creating Fragment

- How to make a Fragment?
  1. Extend Fragment class.
  2. Provide appearance in XML/Java.
  3. Override onCreateView to link the appearance.
  4. Use the Fragment in XML/Java.

2. Just like layout in activity. Each fragment will have separate layout but again not necessary. You can have fragment which is not seen by user so no need of layout.

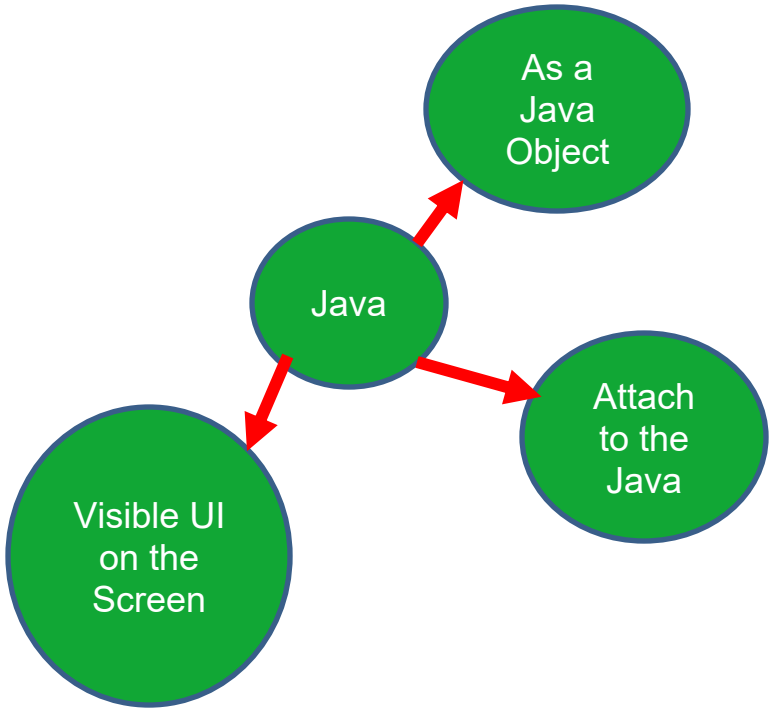
3. This is where you tell the fragment that your appearance is contained in xml or java code.

4. Java is the most preferred approach than XML to create Fragments. The reason is, you create the fragment dynamically at runtime in Java and performs the essential swapping but if you use the fragment in XML, it is statically bound to the user interaction in which you have defined the fragment.

Fragment States

- Different States of Fragment Object.

1. As a Java Object
2. Attach to the Java Object
3. Visible UI on the Screen



Fragment States

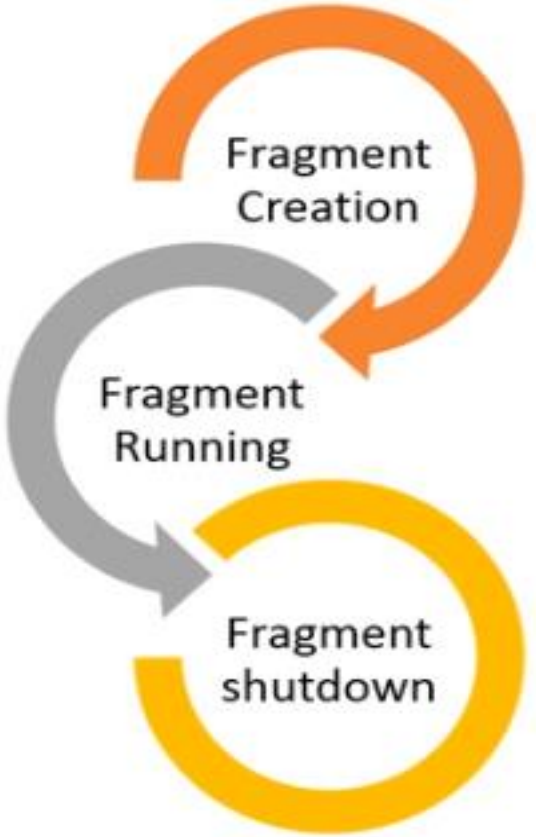
Different States of Fragment Object (*Fragment f=new Fragment()*)

- As a Java Object
  - ✓ It exist in different states.
  - ✓ As a Java Object, it has been not linked with activity but it exist within activity.
- Attach to the Java Object
  - ✓ Object exist and it is link with the activity.
  - ✓ Fragment may or may not be visible to the user.
- Visible UI on the Screen
  - ✓ Object which is fully initialized, attached to the activity and visible to the user.

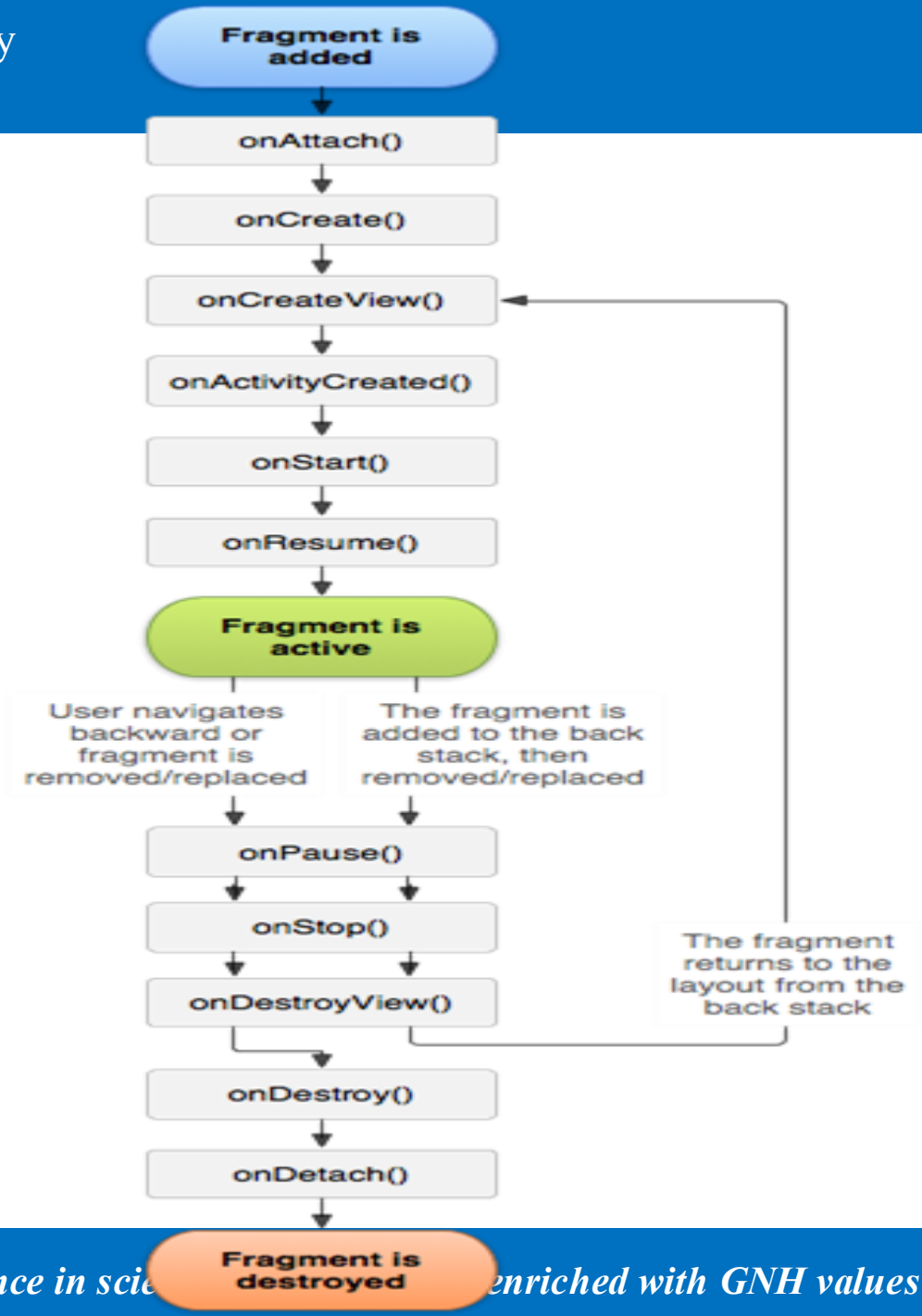
Fragment Lifecycle

- Fragment Lifecycle-Major stages.
- Some of the methods in fragments are responsible for:
  1. Fragment Creation
  2. Fragment Running
  3. Fragment Shutdown

- Lets talk about different methods involved in the creation of Fragments



Fragment

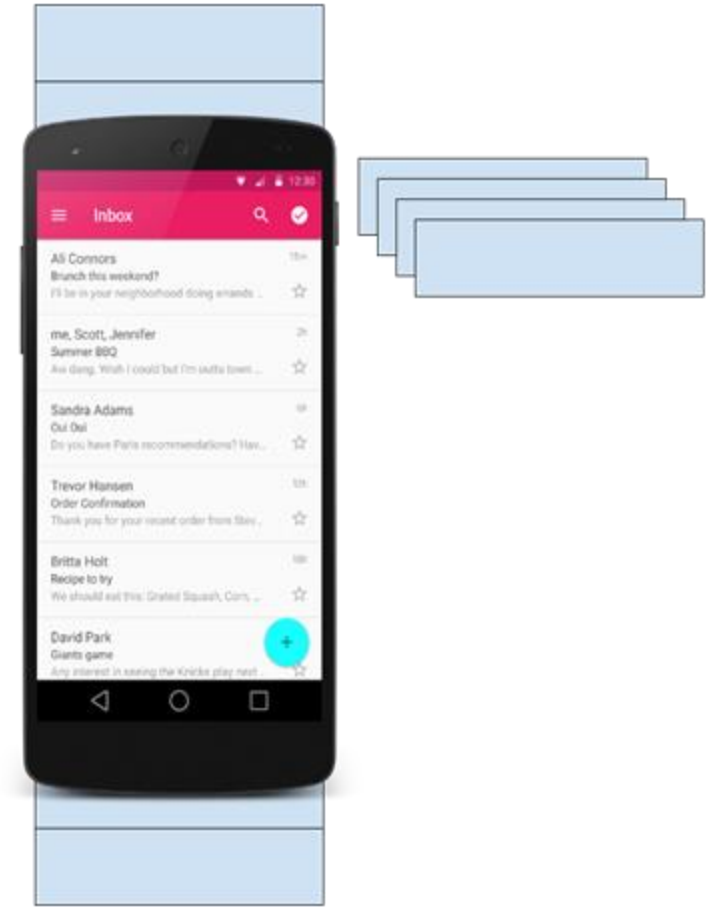

Fragment is added
onAttach()
onCreate()
onCreateView()
onActivityCreated()
onStart()
onResume()
Fragment is active
User navigates backward or fragment is removed/replaced
The fragment is added to the back stack, then removed/replaced
onPause()
onStop()
onDestroyView()
The fragment returns to the layout from the back stack
onDestroy()
onDetach()
Fragment is destroyed

RECYCLER VIEW

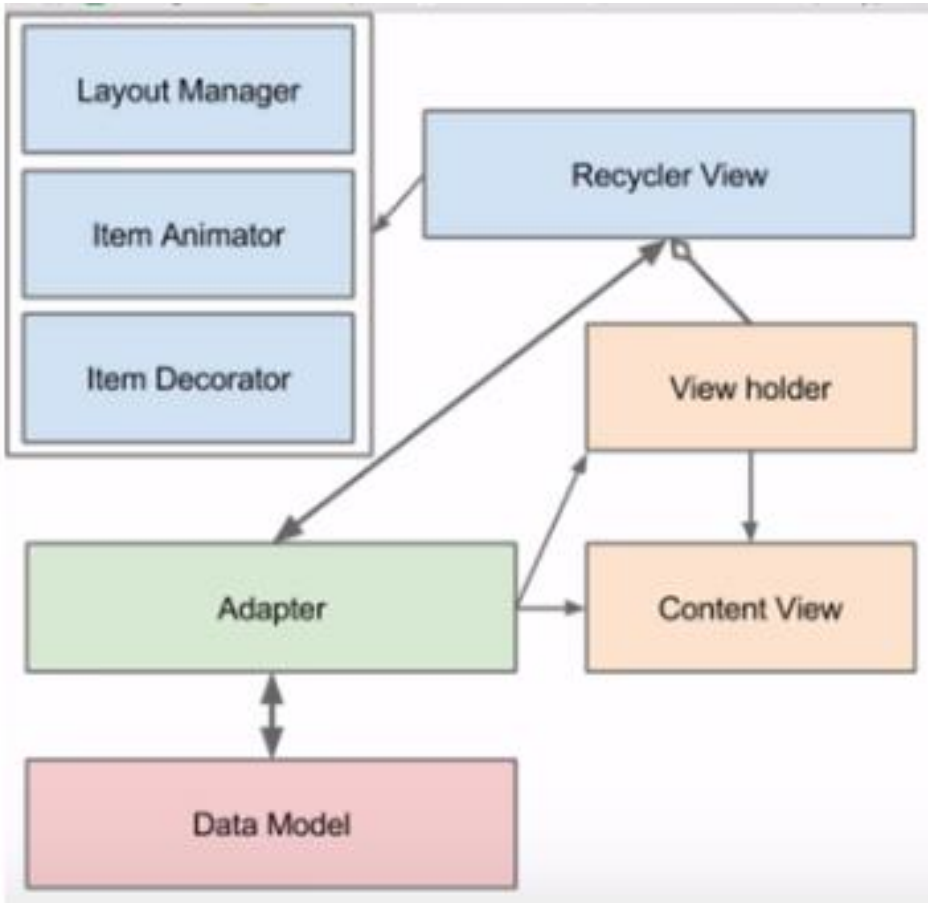
Introduction to Recycler View

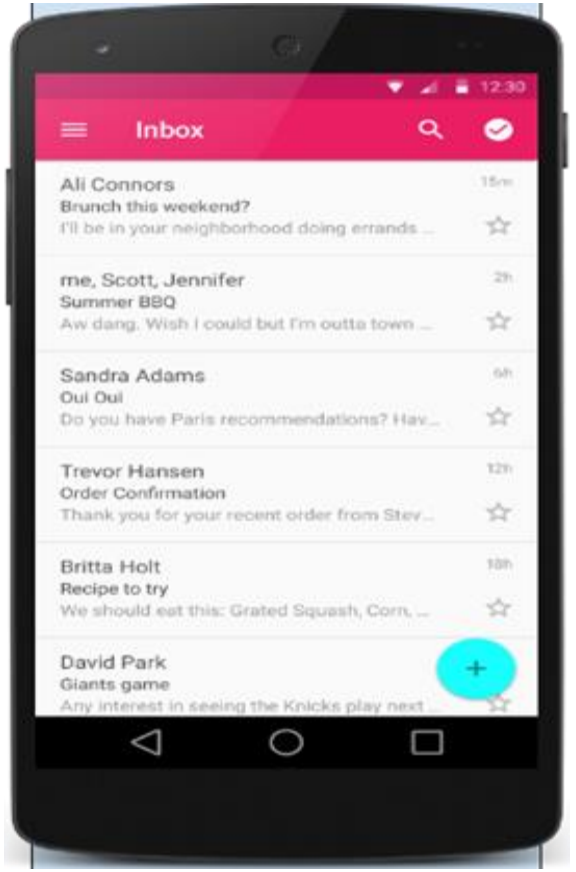
What is a Recycler View?

- The RecyclerView widget is a more advanced and flexible version of ListView.
- It contains large data sets which can be scrolled.
- It is efficient.
- It uses and reuses limited number of views.
- Updates changing data fast.



Architecture of Recycler View (Components fitting together)



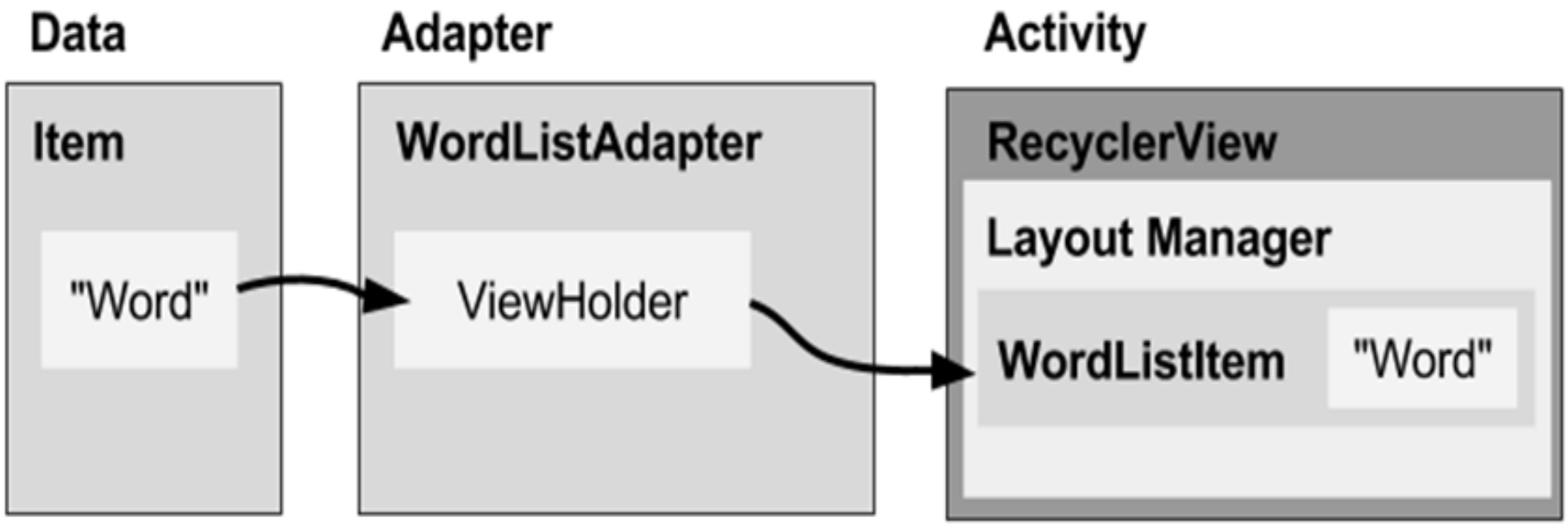


Recycler View Components

The components of RecyclerView

1. Data
2. RecyclerView scrolling list for list items—RecyclerView
3. Layout for one item of data—XML file
4. Layout manager handles the organization of UI components in a view—Recyclerview.LayoutManager
5. Adapter connects data to the RecyclerView—RecyclerView.Adapter
6. View holder has view information for displaying one item—RecyclerView.ViewHolder

Components Fitting Together

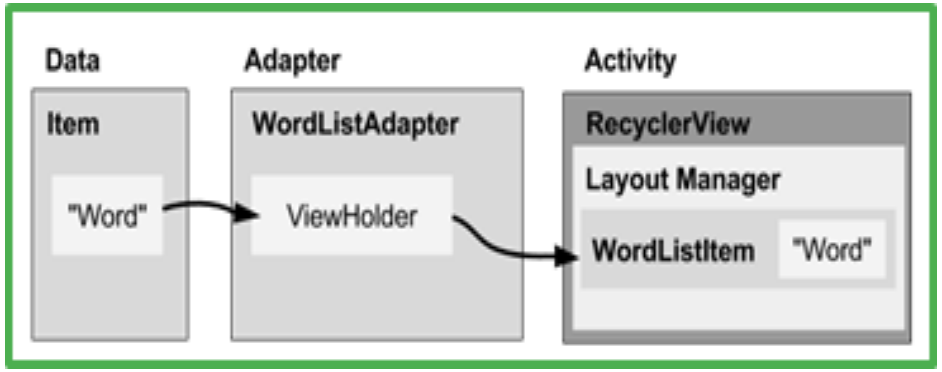


Layout Manager?

- All view groups have layout managers
- Positions item views inside a RecyclerView.
- Reuses item views that are no longer visible to the user
- Built-in layout managers include LinearLayoutManager, GridLayoutManager, and StaggeredGridLayoutManager
- For RecyclerView, extend RecyclerView.LayoutManager

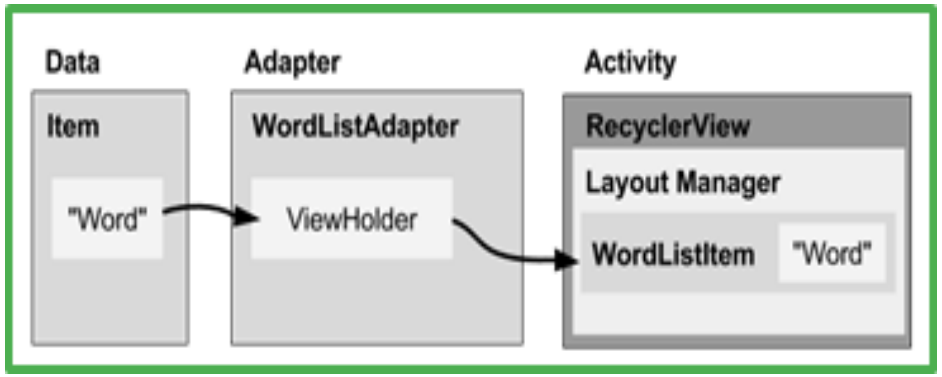
What is Adapter?

- Helps incompatible interfaces work together, for example, takes data from a database Cursor and puts them as strings into a view
- Intermediary between data and view
- Manages creating, updating, adding, deleting item views as the underlying data changes
- RecyclerView.Adapter



What is a View Holder?

- Used by the adapter to prepare one view with data for one list item
- Layout specified in an XML resource file
- Can have clickable elements
- Is placed by the layout manager
- RecyclerView.ViewHolder



Implementing RecyclerView

The steps to create Recycler View

1. Add the RecyclerView dependency to app/build.gradle file.
2. Add RecyclerView to layout.
3. Create XML layout for item.
4. Extend RecyclerView.Adapter.
5. Extend RecyclerView.ViewHolder.
6. In onCreate of activity, create a RecyclerView with adapter and layout manager.

Recycler View

- Add the RecyclerView dependency to app/build.gradle file.

```
dependencies {
   ...
   compile 'com.android.support:recyclerview-v7:24.1.1'
   …
   implementation 'androidx.recyclerview:recyclerview:1.1.0'
}
```

Recycler View

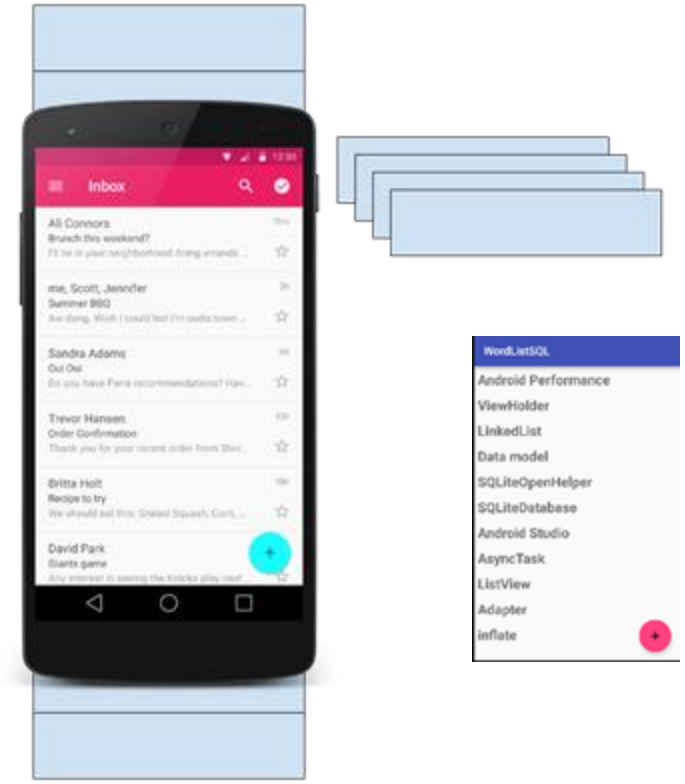
- Add RecyclerView to layout.

```
<android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
</android.support.v7.widget.RecyclerView>
```

Recycler View

- Create XML layout for item.

```
<LinearLayout …>
        <TextView
                android:id="@+id/word"
                style="@style/word_title" />
</LinearLayout>
```



Recycler View

- Extend RecyclerView.Adapter.

```
public class WordListAdapter extends RecyclerView.Adapter
<WordListAdapter.WordViewHolder>
    {
        public WordListAdapter(Context context, LinkedList<String> wordList)     {
                    mInflater = LayoutInflater.from(context);
                    this.mWordList = wordList;
        }
}
```

Recycler View

- Extend RecyclerView.ViewHolder.
- Adapter needs three methods
  1. onCreateViewHolder()
  2. inBindViewHolder()
  3. getItemCount()

Recycler View

- Extend RecyclerView.ViewHolder.
  - onCreateViewHolder()

```
@Override
public WordViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        // Create view from layout
        View mItemView = mInflater.inflate(
             R.layout.wordlist_item, parent, false);
        return new WordViewHolder(mItemView, this);
}
```

Recycler View

- Extend RecyclerView.ViewHolder.
  - inBindViewHolder()

```
@Override
public void onBindViewHolder(WordViewHolder holder, int position) {
        // Retrieve the data for that position
        String mCurrent = mWordList.get(position);
        // Add the data to the view
        holder.wordItemView.setText(mCurrent);
}
```

Recycler View

- Extend RecyclerView.ViewHolder.
  - getItemCount()

```
@Override
public int getItemCount() {
  // Return the number of data items to display
  return mWordList.size();
}
```

Thank you!