Advanced Views

CS 240 – Advanced Programming Concepts

Advanced Views

- RecyclerView
  - For displaying an arbitrarily long repeating list of ViewGroups
  - Examples: Family Map SearchActivity
- ExpandableListView
  - For displaying an expandable lists of items, or multiple expandable lists of items by item group
  - Example: Family Map PersonActivity
- MapView
  - For displaying a map

The Need for An Adapter

The Adapter Pattern

"Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interface"

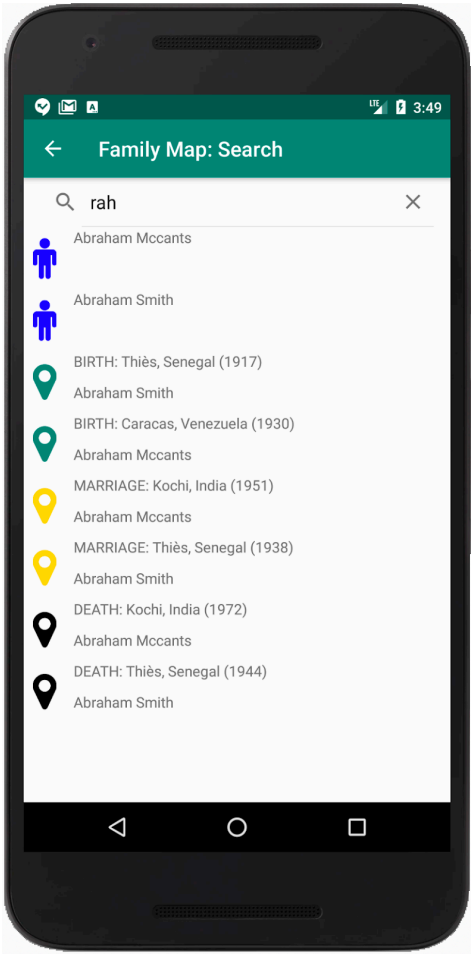
- Design Patterns: Elements of Reusable Object Oriented Software

- Client provides an interface for the adaptee to implement
- Adaptee implements the interface, providing methods the client can call
- Client may also provide an abstract class that implements all or part of the interface that the adaptee can extend

RecyclerView

RecyclerView

- We don’t know how many search results will be returned, but we will repeat a set of Views with a specific layout for each result



RecyclerView

- Makes efficient use of the views for each item by recycling (reusing) them as we scroll through the list
  - We may have a very long list with only a subset visible at any moment
- Needs to make specific requests into our model of people and events to respond to scrolling, etc. happening in the UI
- Needs to be able to request a UI layout for a specific item it needs to display
- Requires the ability to adapt the interface provided by our model to the needs of the view

RecyclerView Layouts

1. Provide a layout for the Activity or Fragment that needs to display the RecyclerView
   - Place a RecyclerView in that layout
2. Provide one or more layouts for the items to be displayed in the RecyclerView
   - One layout if all items to be displayed have the same layout
   - One for each type of item to be displayed if different items have different layouts

RecyclerView: Two Key Classes

ViewHolder

- An implementation of RecyclerView.ViewHolder
- Keeps references to the views being displayed for a specific item to be displayed in the RecyclerView
- Provides a place for you to get references to specific views, add event handlers, etc.

Adapter

- An implementation of RecyclerView.Adapter<ViewHolder>
- Declare the ViewHolder as a generic type
- Interacts with the ViewHolder (creates it as needed)
- Interacts with your model (gets items from it when they need to be displayed) and binds model objects to their Views in the ViewHolder

RecyclerView.Adapter

- Provide a constructor to receive and keep the model objects to be displayed in the RecyclerView
- There are multiple methods you can override, but only 3 you must override:
  - onCreateViewHolder(ViewGroup, int viewType)
    - Ignore viewType unless you override other methods to support multiple types of views
  - onBindViewHolder(VH holder, int position)
    - Get the item at 'position' and pass it to a 'bind' method in ViewHolder
  - getItemCount()
    - Return the total number of items to be displayed in the RecyclerView
- Example: CriminalIntent-Chapt10/CrimeListFragment.CrimeAdapter

RecyclerView.ViewHolder

- Provide a constructor that does the following:
  - Inflates the view for the item to be displayed and passes it to the parent constructor
  - Sets itself as an onClick listener if you want to be able to click items in the RecyclerView
  - Gets references to the UI items that will need to be manipulated to display data
- Use the itemView variable inherited from parent to interact with the views

RecyclerView.ViewHolder

- Provide a method that can be called by the adapter's `onBindViewHolder(…)` method to set the values in the View to display an item
- Implement the View.OnClickListener interface and provide an `onClick(…)` method if you want to be able to click items in the RecyclerView
- Example: CriminalIntent-Chapt10/CrimeListFragment.CrimeHolder

RecyclerView Setup: Library Dependency

- Add the RecyclerView library to your project
  - See Android book pg. 171 - **Menu options and library version have changed from book instructions. Follow these steps:**
    - Open Project Structure dialog
    - Select Dependencies
    - Select the “app” module
    - Click plus under “Declared Dependencies” (not the one under Modules)
    - Select “Library Dependencies”
    - Search for ”recyclerview” (all lowercase)
    - Select the latest release version of “androidx.recyclerview”

RecyclerView Setup: xml and .java File

- Wherever the book references "android.support.v7.widget.RecyclerView" use "**androidx.recyclerview.widget.RecyclerView**" instead
- Do the following in the activity or fragment displaying the RecyclerView:
  - Get a reference to the RecyclerView in the layout and set a layout manager (see CrimeListFragment.onCreateView(…)
  - Create an instance of the RecyclerView.Adapter, passing it the data to be displayed

Supporting Multiple View Types in a RecylerView

- Create two layouts (one for each view type)
- Override `int getItemViewType(int position)` in Adapter
- Create two bind methods in ViewHolder (one for each type of object)
- Inflate the appropriate layout in `onCreateViewHolder` based on the viewType passed in
- Call the appropriate bind method in `onBindViewHolder`
- Example: RecyclerViewExample_MultitpleViewTypes

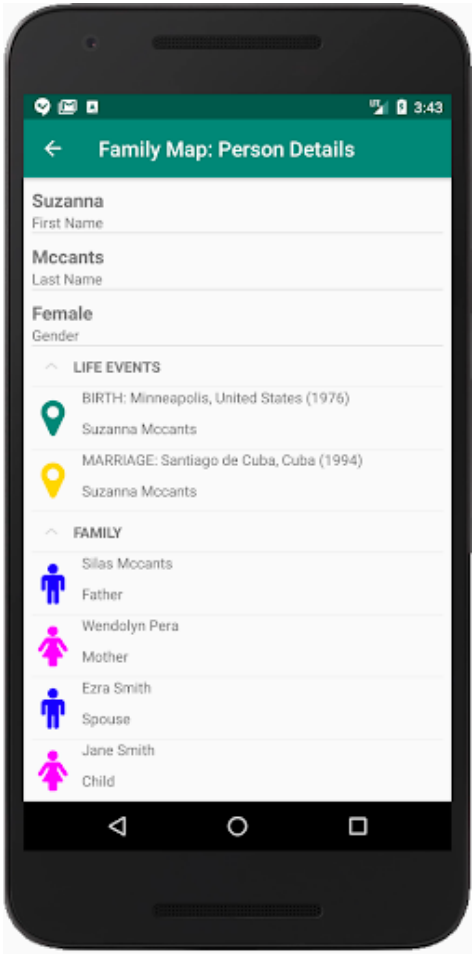
Family Map Search Activity

- Your RecyclerView will display People and Events matching the search criteria
- Both can use the same RecyclerView item layout, with the second detail line being blank for people
- Keep separate person and event lists in your adapter and have two bind methods in your view holder (one for people and one for events)

ExpandableListView

ExpandableListView

- You could do this with a RecyclerView with four different view types, but ExpandableListView is a better fit.
- Requires an adapter (like RecyclerView) but no ViewHolder.



ExpandableListView Layouts

1. Provide a layout for the Activity or Fragment that needs to display the ExpandableListView.
   - Place an ExpandableListView in that layout
2. Provide one or more layouts for the groups to be displayed (group headings)
   - You can usually use one layout for all groups
3. Provide one or more layouts for the items to be displayed within the groups
   - You can only use one layout for all items if the layouts for the items are the same across groups

ExpandableListView Adapter

- Subclass BaseExpandableListAdapter
- Provide a constructor to receive and keep the model objects to be displayed in the ExpandableListView
- There are more methods to override than for RecyclerView
- You do not use a ViewHolder
- Set the adapter for the ExpandableListView in onCreate(Bundle)
- Example: ExpandableListViewExample.zip

Google Maps

Working with Google Maps

- Multiple ways to work with maps in an Android application
  1. Create a Map Activity
  2. Place a MapView in a layout and use it to load a map into an activity or fragment
     - Requires you to override lifecycle callbacks and pass them on to the MapView
  3. Create a SupportMapFragment

- Only options 2 and 3 allow you to create a fragment with a map on it (so option 1 is not available for FamilyMapClient)
- All three options start by adding a Map Activity to your project

Creating a Support Map Fragment

1. Add a Google Maps Activity to your application (File -> New -> Google -> Google Maps Activity)
2. Get your google_maps_key and copy it into google_maps_api.xml
   - See instructions in values/google_maps_api.xml
3. Delete the maps activity and corresponding layout (you only needed it to set your application up to support maps)
4. Create an empty fragment (and delete the code Android Studio places in it)
5. Create the layout for your map fragment, and include a SupportMapFragment
6. Write code in your fragment's .java file to display your map

- Example: MapFragmentExample.zip

5. Creating a Layout for a Map Fragment

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

   <fragment
      android:id="@+id/map"
      class="com.google.android.gms.maps.SupportMapFragment"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_above="@id/mapTextView"/>

   <TextView
      android:id="@+id/mapTextView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="8dp"
      android:text="@string/mapFragmentMessage"
      android:layout_alignParentBottom="true"
      android:textAlignment="center"/>

</RelativeLayout>
```

6. Fragment Code to Display a Map

```
public class MapFragment extends Fragment implements OnMapReadyCallback {
   private GoogleMap map;

   @Override
   public View onCreateView(@NonNull LayoutInflater layoutInflater, ViewGroup container, Bundle savedInstanceState) {
      super.onCreateView(layoutInflater, container, savedInstanceState);
      View view = layoutInflater.inflate(R.layout.fragment_map, container, false);

      SupportMapFragment mapFragment = (SupportMapFragment) getChildFragmentManager().findFragmentById(R.id.map);
      mapFragment.getMapAsync(this);

      return view;
   }

   @Override
   public void onMapReady(GoogleMap googleMap) {
      map = googleMap;

      // Add a marker in Sydney and move the camera
      LatLng sydney = new LatLng(-34, 151);
      map.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
      map.animateCamera(CameraUpdateFactory.newLatLng(sydney));
   }
}
```

Useful Map Related Methods

- GoogleMap.clear()
- GoogleMap.addMarker(…)
- GoogleMap.addPolyLine(…)
- GoogleMap.moveCamera(…)
- GoogleMap.animateCamera(…)
- GoogleMap.setMapType(…)
- GoogleMap.setOnMarkerClickListener(Marker)
- Marker.setTag(…)
- Marker.getTag(…)