In-Class Homework #7 – Android RecyclerView

CUS 357 – Winter 2017
Due Date: End of class, Mar 24, 2017

Learning Objectives

- Familiarity with Android's RecyclerView and related concepts.
- Familiarity with Android Fragments.

Please work in pairs to get as much of this assignment done as possible by the end of today's class session.

Overview

Begin this exercise using the finished product of your In-Class Homework #4 solution (e.g. your stylized Android GeoCalculator). If you did not complete that assignment, you are welcome to use the instructor's solution (available on Blackboard with this assignment) as your starting point. If you use your own code, keep in mind there might be some minor changes in the code snippets below, depending on you named things. It shouldn't be too difficult to adjust if needed.

Using what you have learned in Lecture 12 you will be using RecyclerView and related inner classes, as well as Fragments to add a "history" feature to your Android GeoCalculator app as shown in Figure 1. To help you get this done in 50 minutes, we will guide you through the coding in a step-by-step fashion.

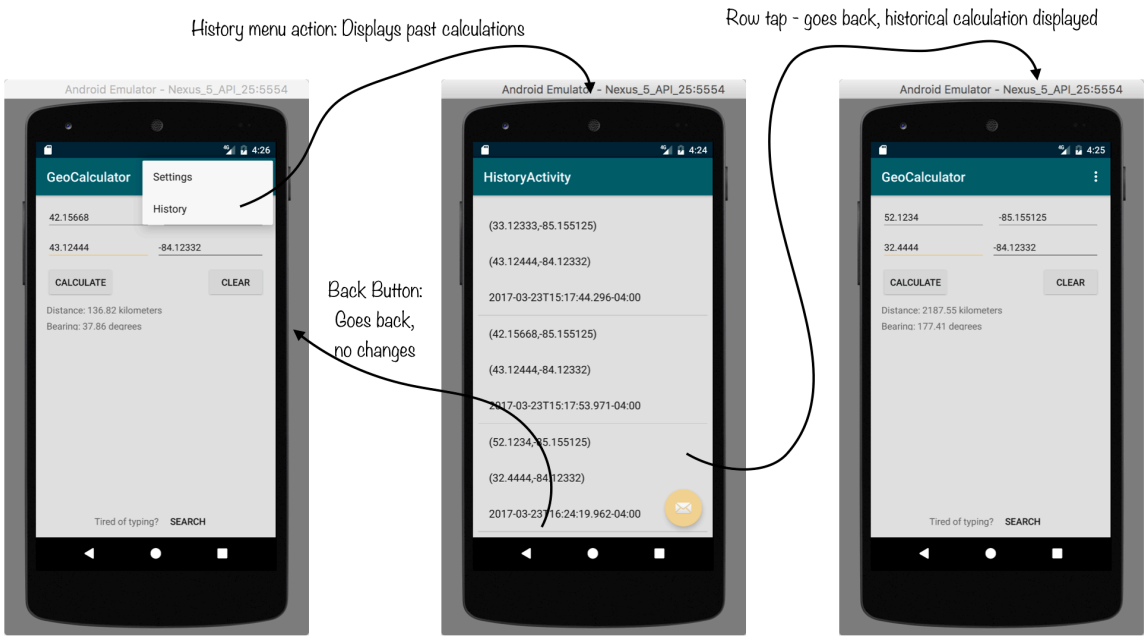

Figure 1. Adding a "history" feature using RecyclerView.

Each time the calculate button is pressed with valid inputs, a time stamped entry will be recorded for historical purposes. When the user access the history action on the menu, a RecyclerView

populated with all of the history to-date will be displayed. If the user hits the back button, the history activity popped, and the user is returned to the calculator screen with no changes. However, if the user taps an entry in the history activity, the activity is popped, and the user is returned to the calculator screen with that historical calculation repopulated in the text views and the distance and bearing computed and displayed below.

Introducing the Fragment and RecyclerView

We will be using RecyclerView from the Android support libary.

Step 1: Add the following dependency to your gradle build dependencies (build.gradle – Module: app):

compile 'com.android.support:recyclerview-v7:x.y.z'

Note the x.y.z needs to be updated per your development environment. Next we need to add a new activity that will host our historical calculations.

Step 2: Introduce a new Basic Activity called HistoryActivity that will host your list of historical calculations. Select the Java source package in the project explorer, right click and go to New > Activity > Basic Activity. In the dialog make sure the activity name is HistoryActivity and leave the other fields at their default values.

Step 3: Add a new fragment by selecting the Java source package, right click and go to New > Fragment > Fragment (List). Leave all the values on the dialog as default before continuing.

Step 4: Refactor > Rename the following four entities that were generated by the previous step:

ItemFragment → HistoryFragment
MyItemRecyclerViewAdaptor → HistoryAdaptor
fragment_item.xml → fragment_history.xml
fragment_item_list.xml → fragment_history.xml

Step 5: Open up content_history.xml (the layout for HistoryActivity) in the layout editor, and drag a fragment into the layout. From the dialog, that pops up be sure to select the "HistoryFragment" entry. Dismiss the warning messages (we know what we're doing…).

Step 6: In the XML editor, make sure the width and height of the fragment you just added is set to match_parent.

We are almost ready to try this new functionality out, but first we need to add the new activity to the menu of our geo calculator.

Step 7: Add a new item to menu/main_menu.xml for the history feature.

```
<item android:title="History"
    android:id="@+id/action_history"
    app:showAsAction="never" />
```

Step 8: Wire up the new menu item in MainActivity by updating the onOptionsItemSelected() method.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if(item.getItemId() == R.id.action_settings) {
        Intent intent = new Intent(MainActivity.this,
                MySettingsActivity.class);
        startActivityForResult(intent, SETTINGS_RESULT );
        return true;
    } else if(item.getItemId() == R.id.action_history) {
        Intent intent = new Intent(MainActivity.this, HistoryActivity.class);
        startActivityForResult(intent, HISTORY_RESULT );
        return true;
    }
    return false;
}
```

Step 9: Since HistoryActivity will be hosting HistoryFragment, let's wire it up to listen for the row tap events. Make sure HistoryActivity implements the HistoryFragment and then implement the method:

```
 public class HistoryActivity extends AppCompatActivity
            implements HistoryFragment.OnListFragmentInteractionListener  {

            .
            .
            .
       @Override
       public void onListFragmentInteraction(DummyContent.DummyItem item) {
           System.out.println("Interact!");
       }
}
```

Step 10: At this point you should be able to run and check that the generated RecyclerView code works correctly.

Customizing the Historical Data Being Remembered

Android Studio generated a generic Item type in the placeholder code. Let's customize that to handle our historical calculation data types and cause our calculator to store history in that type.

Step 11: Refactor > Rename the following generated classes:

DummyItem → History (inner class in DummyContent)
DummyContent → HistoryContent

Step 12: Add the Joda Time dependencies to your gradle build config:

```
compile 'net.danlew:android.joda:2.9.5'
```

Step 13: Initialize Joda time by introducing this class (as in the Screencast). **Don't forget to add the class to your AndroidManifest.xml with the application attribute android:name=".GeoCalculatorApp".**

```
public class GeoCalculatorApp extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        JodaTimeAndroid.init(this);
    }
}
```

Step 14. Rewrite HistoryContent and its inner class as follows:

```
public class HistoryContent {
    public static final List<HistoryItem> ITEMS = new ArrayList<HistoryItem>();

    private static void addItem(HistoryItem item) {
        ITEMS.add(item);
    }

    public static class HistoryItem {
        public final String origLat;
        public final String origLng;
        public final String destLat;
        public final String destLng;
        public final DateTime timestamp;

        public HistoryItem(String origLat, String origLng, String destLat,
                           String destLng, DateTime timestamp) {
            this.origLat = origLat;
            this.origLng = origLng;
            this.destLat = destLat;
            this.destLng = destLng;
            this.timestamp = timestamp;
        }

        @Override
        public String toString() {
            return "(" + this.origLat + "," + this.origLat + ")";
        }
    }
}
```

Step 15: Let's add a decorator to separate our vertical stack of entries. See you can figure out where you might do this… here is the code:

```
DividerItemDecoration did = new DividerItemDecoration(recyclerView.getContext(),
        DividerItemDecoration.VERTICAL);
recyclerView.addItemDecoration(did);
```

Step 17: In your fragment_history.xml let's tweak the row layout as follows:

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

    <TextView
        android:id="@+id/p1"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/text_margin"
        android:textAppearance="?attr/textAppearanceListItem" />

    <TextView
        android:id="@+id/p2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/text_margin"
        android:textAppearance="?attr/textAppearanceListItem" />
    <TextView
        android:id="@+id/timestamp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/text_margin"
        android:textAppearance="?attr/textAppearanceListItem" />

</LinearLayout>
```

Step 16: In your adaptor implementation redefine your ViewHolder inner class to work with the new layout:

```
public class ViewHolder extends RecyclerView.ViewHolder {
    public final View mView;
    public final TextView mP1;
    public final TextView mP2;
    public final TextView mDateTime;
    public HistoryItem mItem;

    public ViewHolder(View view) {
        super(view);
        mView = view;
        mP1 = (TextView) view.findViewById(R.id.p1);
        mP2 = (TextView) view.findViewById(R.id.p2);
        mDateTime = (TextView) view.findViewById(R.id.timestamp);
    }

    @Override
    public String toString() {
        return super.toString() + " '" + mDateTime.getText() + "'";
    }
}
```

Step 17: Your binding code also needs to be adjusted to handle the new layout:

```
@Override
public void onBindViewHolder(final ViewHolder holder, int position) {
    holder.mItem = mValues.get(position);
    holder.mP1.setText("(" + holder.mItem.origLat + "," + holder.mItem.origLng
+ ")");
    holder.mP2.setText("(" + holder.mItem.destLat + "," + holder.mItem.destLng
+ ")");
    holder.mDateTime.setText(holder.mItem.timestamp.toString());

    holder.mView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (null != mListener) {
                // Notify the active callbacks interface (the activity, if the
```

```
                // fragment is attached to one) that an item has been selected.
                mListener.onListFragmentInteraction(holder.mItem);
            }
        }
    });
}
```

Step 18: Update the fragment listener method in the HistoryActivity to return the selected value to the MainActivity:

```
public void onListFragmentInteraction(HistoryContent.HistoryItem item) {
    System.out.println("Interact!");
    Intent intent = new Intent();
    String[] vals = {item.origLat, item.origLng, item.destLat, item.destLng};
    intent.putExtra("item", vals);
    setResult(MainActivity.HISTORY_RESULT,intent);
    finish();
}
```

Step 19: Update the MainActivity to receive this result from the HistoryActivity:

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == SETTINGS_RESULT) {
        this.bearingUnits = data.getStringExtra("bearingUnits");
        this.distanceUnits = data.getStringExtra("distanceUnits");
        updateScreen();
    } else if (resultCode == HISTORY_RESULT) {
        String[] vals = data.getStringArrayExtra("item");
        this.p1Lat.setText(vals[0]);
        this.p1Lng.setText(vals[1]);
        this.p2Lat.setText(vals[2]);
        this.p2Lng.setText(vals[3]);
        this.updateScreen();  // code that updates the calcs.
    }
}
```

Step 20: Go ahead and run your app at this point, and make sure you can scroll through your history and select one, returning to the main screen with the fields filled correctly.

If you did everything correctly, you should now be done! Do a quick demo for the instructor, and then have one of the pair post a zipped solution (or github url) to Blackboard. If you do not finish, post the project to Blackboard with a sentence or two description of what you finished.