Amazon Maps

See FAQ on web site.

Steps:

1. Add permissions to your app's AndroidManifest.xml file
2. Download Amazon Maps SDK and integrate it into your project
3. Register your app with Amazon, and register your development machines so they can use the Maps API
4. Embed Amazon Maps fragment in your UI, and use the Amazon Maps API in your code

- Google Maps is not available for Amazon Devices. Instead, you will need to use Amazon Maps
- Update the `AndroidManifest.xml` to include within the `manifest` tag:
  - `<uses-permission android:name="android.permission.INTERNET" />`
  - `<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />`
  - `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />`
  - `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />`

- Download the Amazon Apps & Games Services SDK for Fire OS and Android here
- Unzip the Amazon-Android-SDKs.zip file in a location of your choice.
- In your Android Studio project, create a new module by selecting `File -> New -> New Module`
- Select `Import .JAR/.AAR Package` and click Next
- In the "File name" field, browse to the location where you unzipped Amazon-Android-SDKs.zip, and select the file `Amazon-Android-SDKs/AmazonMaps/2.3/amazon-maps-api-v2.aar` and click Finish
- In your project's settings.gradle file, include the ':amazon-maps-api-v2' module (if it is already there, do not add it again). It should look something like this:

```
include ':app', ':amazon-maps-api-v2'
```

- In the "app" module's build.gradle file, add the following

```
dependencies {
    ....
    compile project(':amazon-maps-api-v2')
}
```

- Gradle will need to resync, typically Android Studio will ask if you want to do this

- In order to use Amazon Maps fully, you will need to register your app with Amazon to test it
- You will need to get the MD5 hash of the project
- To do this, run the follwoing command in the terminal
  - **Mac OS/ Linux**: `keytool -v -list -alias androiddebugkey -keystore ~/.android/debug.keystore -storepass android`
  - **Windows**: `keytool -v -list -alias androiddebugkey -keystore %HOMEPATH%\.android\debug.keystore -storepass android`
- The MD5 has will be under `Certificate fingerprints`
- Go to the Amazon Mobile Apps & Games Developer Portal and sign-in or create an account
- In the dashboard, click `Add a New App`
- Fill out the required information
- Click `Maps` and then `Add a Debug Registration`
- fill in the package name and developer signature which is the MD5 hash and click `Submit`
- It may take up to an hour for these changes to propagate through Amazon's system
- If you program on your laptop and on the open lab machines, you will need to `Add a Debug Registration` for both. In other words, you will need to register the MD5 hash for your laptop and register the MD5 hash for the open labs. (All of the open lab machines have the same MD5 hash, so registering one of them will register all of them.) This way, Amazon Maps should work no matter where you run your app from (laptop or open lab machine).

- You should now be able to follow the steps here to use Amazon Maps

- In your Map Fragment layout, embed the Amazon Maps fragment:

```
<fragment android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:id="@+id/map"
    tools:context="edu.byu.cs240.familymap.ui.MapActivity"
    android:name="com.amazon.geo.mapsv2.SupportMapFragment" />
```

- In your Map Fragments onCreateView method:

```
map = ((SupportMapFragment)getChildFragmentManager().findFragmentById(R.id.map)).getMap();
map.setMapType(…);
map.setOnMarkerClickListener(…);
```

- Other useful methods:
  - Map.clear(…)

- Map.addMarker(…)
- Map.addPolyLine(…)
- Map.moveCamera(…)
- Map.getCameraPosition(…)

RecyclerView

RecyclerView can be used to display collections of items in different layouts (vertical list, horizontal list, grid, etc.)

Collections can be very large. It is not feasible to create an item view for each collection item. Only a small number of items are visible at any moment. Idea: create only the number of item views that are simultaneously visible. As the user scrolls, re-bind those few item views to the currently-visible items (i.e., recycle the item views).

In CriminalIntent(10), explain CrimeListFragment class's use of RecyclerView

Family Map Application

In Family Map, the Person, Filter, and Search activities all have dynamic lists that can be implemented using Android's RecyclerView.

Filter Activity and Search Activity should be implemented using the built-in RecyclerView.Adapter class. Person Activity is more complex, because the list has two expandable sub-sections (events and family members). There are two approaches to implementing this dynamic list: 1) Use the regular RecyclerView.Adapter (like Filter Activity and Search Activity), or 2) Use Big Nerd Ranch's ExpandableRecyclerAdapter class, which extends RecyclerView.Adapter with support for expandable lists.

Use Big Nerd Ranch's ExpandableRecyclerAdapter Class

The easiest option is to use the Big Nerd Ranch ExpandableRecyclerAdapter class. See the tutorial and code links at the following URL:

https://www.bignerdranch.com/blog/expand-a-recyclerview-in-four-steps/

Use RecyclerView.Adapter Directly (without help from ExpandableRecyclerAdapter)

RecyclerView.Adapter provides no special support for expandable lists, so you must roll your own (this is why the previous option is easier).

Four types of list items:

1. Events header
2. Event items
3. Family header
4. Family items

Four corresponding "view holder" classes:

1. EventsHeaderHolder

    a. Expanded and collapsed icons
    b. OnClickListener changes header icon and notifies RecyclerView of data change
        i. Calls RecylcerView.notifyItemRangeRemoved OR RecylcerView.notifyItemRangeInserted
2. FamilyHeaderHolder
    a. Expanded and collapsed icons
    b. OnClickListener changes header icon and notifies RecyclerView of data change
        i. Calls RecylcerView.notifyItemRangeRemoved OR RecylcerView.notifyItemRangeInserted
3. EventHolder
    a. Binds event properties to widgets
    b. OnClickListener starts MapActivity centered on clicked event
4. PersonHolder
    a. Binds person properties to widgets
    b. OnClickLisitener starts PersonActivity displaying clicked person

ADAPTER CLASS

Override getItemCount():

```
itemCount = 2;
if (events expanded)
        itemCount += number of events
if (family expanded)
        itemCount += number of family members
```

Override getItemViewType(): Return value indicating each item's "view type"

Override onCreateViewHolder(): Return view holder of requested type

Override onBindViewHolder(): Binds item data into item view (handles four different view types)

1. EventsHeaderHolder
    a. Set header icon (expanded or collapsed)
2. FamilyHeaderHolder
    a. Set header icon (expanded or collapsed)
3. EventHolder
    a. Copy event data into widgets
4. PersonHolder
    a. Copy person data into widgets