AM's workflow for having Google Maps in Android Apps

Here is my pattern for having Google Maps in Android Apps.

Administrative stuff and relevant URLs

First, read about "billing" at
https://console.cloud.google.com/billing/

Billing must be enabled to use the "Google Maps" platform:
https://console.cloud.google.com/project/_/billing/enable

Using a billing account, visit the URL:
https://console.cloud.google.com/google/maps-apis/credentials

To configure the settings of a project:
https://console.cloud.google.com/admin/settings?project=<project name>

To check API usage for a project:
https://console.cloud.google.com/google/maps-apis/metrics?project=<project name>

Key for the API "Maps SDK for Android"

I created a new API Key for a "ddm231118" named Android app and got:
AIza…Iuoc (key not fully shown, for security reasons)

Then, I restricted it:
- to "Android Apps";
- to the API "Maps SDK for Android".

To restrict to "Android Apps", one has to add a "new item", consisting of:
- package name;
- SHA-1 certificate fingerprint.

The easiest way to get those values is, AFTER having created an Android Studio project, to use Gradle's Android task "signing report": from Android Studio's terminal run the command `./gradlew signingReport`

In my case:
Package = "com.joythis.android.ddm231128";
SHA-1 certificate fingerprint: "A0:0E:2F:…:46:66"

The full task's output was:

```
./gradlew signingReport

Welcome to Gradle 8.2!

Here are the highlights of this release:
 - Kotlin DSL: new reference documentation, assignment syntax by default
 - Kotlin DSL is now the default with Gradle init
 - Improved suggestions to resolve errors in console output

For more details see https://docs.gradle.org/8.2/release-notes.html

Starting a Gradle Daemon, 2 incompatible Daemons could not be reused, use --status for details

> Task :app:signingReport
Variant: debug
Config: debug
```

```
Store: C:\Users\research\.android\debug.keystore
Alias: null
----------
Variant: debugAndroidTest
Config: debug
Store: C:\Users\research\.android\debug.keystore
Alias: AndroidDebugKey
MD5: 68:...:0A
SHA1: A0:...:46:66
SHA-256: 2D:...:08
Valid until: Thursday, September 28, 2045
----------

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from
your own scripts or plugins.

For more on this, please refer to
https://docs.gradle.org/8.2/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle
documentation.

BUILD SUCCESSFUL in 8s
1 actionable task: 1 executed
```

The official documentation for the "maps" platform is at:
https://developers.google.com/maps/documentation/android-sdk/config

Since 2020, to use the maps API, it became necessary to have "billing enabled".
By using the URL
https://console.cloud.google.com/project/_/billing/enable
one can select a project to which to apply "billing", via a "billing account".

These URLs can also be helpful:
https://console.cloud.google.com
https://accounts.google.com
https://console.cloud.google.com/google/maps-apis/credentials
https://console.cloud.google.com/apis/dashboard
https://console.cloud.google.com/google/maps-apis/apis/maps-android-backend.googleapis.com/

So:
#1) enable billing on the new project;

#2) enable the "Maps SDK for Android" API on the project;
https://console.cloud.google.com/apis/library/maps-android-backend.googleapis.com

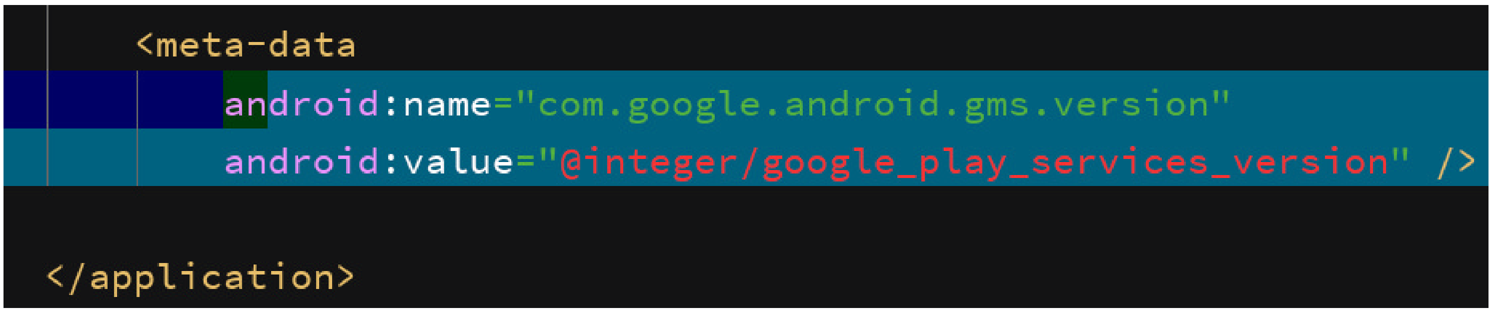
Metrics will be available here:
https://console.cloud.google.com/google/maps-apis/metrics?project=<project name>

#3) get and restrict an API key
https://developers.google.com/maps/documentation/android-sdk/get-api-key

#4) edit "AndroidManifest.XML" and add the following as content of the application node:

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

This will cause a "red" syntax moment - do not worry about it, as it will disappear once a still missing dependency is automatically added to the project:



```
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />

</application>
```

Also request the "external storage permission", although is not needed for Google Play Services SDK >=8.3

```
<uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

and also

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

and

```
<uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>
```

#5) The project will require a dependency entry for "play-services-maps". The simplest way to do this, is to edit any XML layout in the project, and add it a "Google" "MapView" object, only to delete the object [immediately] after, just to have the file "build.gradle (Module: <project name>.app)" to refer the proper library, e.g.:

```
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

Here is one possible dependencies node:

```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.android.gms:play-services-maps:17.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Finally, the API key previously obtained must be explicitly associated with the project.

Input the API key as meta-data, directly in the Manifest, inside the application node:

```
<meta-data
       android:name="com.google.android.geo.API_KEY"
        android:value="@string/google_maps_key" />
```

The full AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.joythis.android.a201214_rmaps">

    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.201214_rmaps">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="AI…oc"/>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <!--
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="${mapsApiKey}" />

        -->

    </application>

</manifest>
```

Layouts for the GoogleMap

First, create the simplest possible layout with a "Google Map" fragment.

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

    <fragment
        android:id="@+id/idGMapFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        class="com.google.android.gms.maps.MapFragment"/>

</LinearLayout>
```

Then, include it in the layout of the Activity where a map is needed.
For example:

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

    <Spinner
        android:id="@+id/idSpnMapType"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"></Spinner>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <EditText
            android:id="@+id/idEtLat"
            android:layout_weight="1"
            android:inputType="numberDecimal|numberSigned"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <EditText
            android:id="@+id/idEtLong"
            android:layout_weight="1"
            android:inputType="numberDecimal|numberSigned"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
        <Button
            android:id="@+id/idBtnGo"
            android:text="Go!"
            android:layout_weight="1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>

    <include
        layout="@layout/googlemap_ll"></include>

</LinearLayout>
```

AmGoogleMap, an helper class

To help me working with GoogleMaps, I created an auxiliary class named AmGoogleMap.

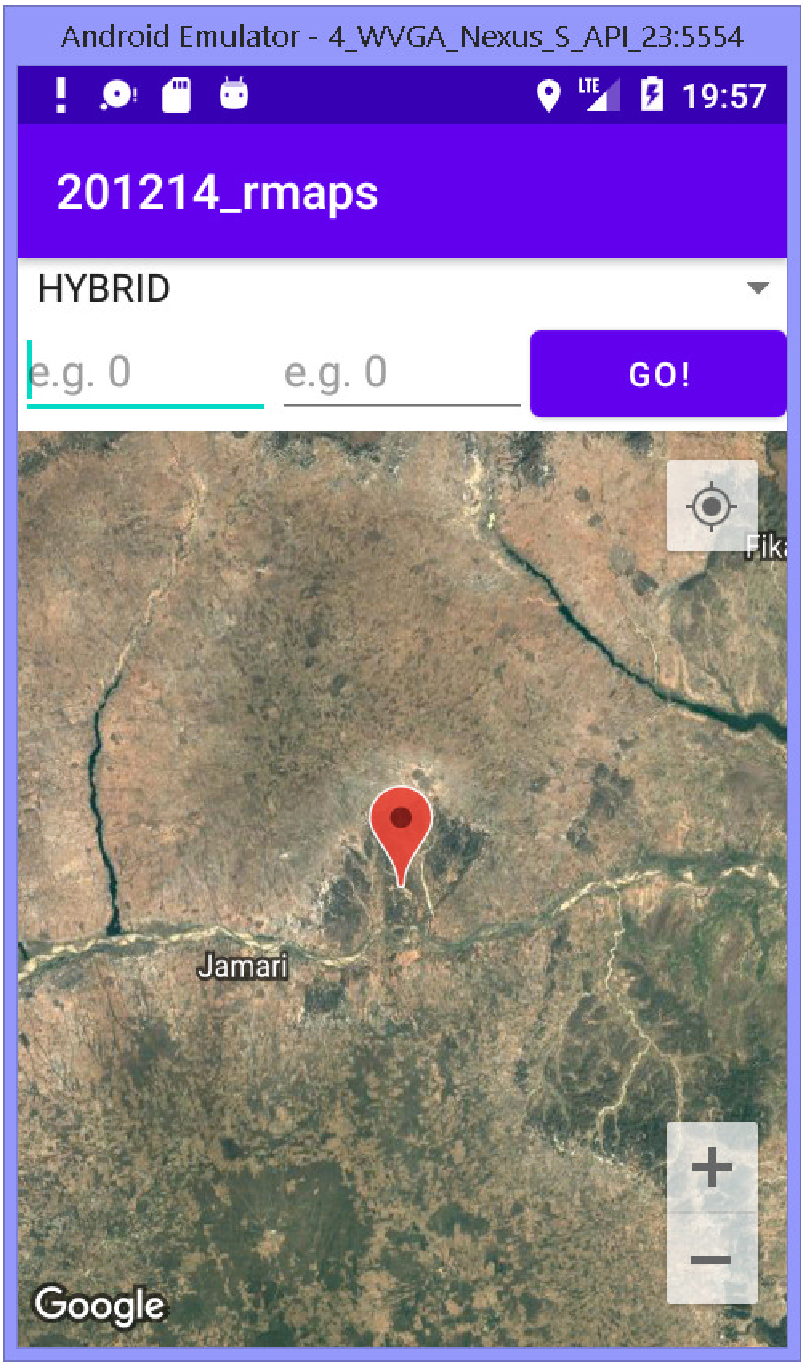
```
package com.joythis.android.a201214_rmaps;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

/**
 * Created by Artur Marques on 2016-11-10.
 */
public class AmGoogleMap implements OnMapReadyCallback {
    android.app.Activity mActivity;
    GoogleMap mGoogleMap;

    AmUtil mUtil;
    int mAppRequestCode;

    public final static String[] NECESSARY_PERMISSIONS = {
        android.Manifest.permission.ACCESS_COARSE_LOCATION,
        android.Manifest.permission.ACCESS_FINE_LOCATION
    };

    //-----------------------------------------------------------------------------------------------------------
--------
    public AmGoogleMap(
            android.app.Activity pA,
            int pAppRequestCode
    )
    {
        mAppRequestCode = pAppRequestCode;
        mActivity = pA;
        mUtil = new AmUtil(mActivity);
    }//MeuMapa

    //----------------------------------------------------------------------------------------------------------
    public Marker addMarkerToMap(
            LatLng pSomeLocationAsLatLong,
            String pSomeDescription
    )
    {
        //https://developers.google.com/android/reference/com/google/android/gms/maps/model/MarkerOptions
        MarkerOptions markerOptions = new MarkerOptions();


//https://developers.google.com/android/reference/com/google/android/gms/maps/model/MarkerOptions.html#posit
ion(com.google.android.gms.maps.model.LatLng)
        //Sets the location for the marker.
        markerOptions.position (pSomeLocationAsLatLong);


//https://developers.google.com/android/reference/com/google/android/gms/maps/model/MarkerOptions.html#title
(java.lang.String)
        //Sets the title for the marker.
        String strTitle="@"+pSomeLocationAsLatLong.latitude+", "+pSomeLocationAsLatLong.longitude;
        markerOptions.title (strTitle);


//https://developers.google.com/android/reference/com/google/android/gms/maps/model/MarkerOptions.html#snipp
et(java.lang.String)
        //Sets the snippet for the marker.
        markerOptions.snippet(pSomeDescription);


//https://developers.google.com/android/reference/com/google/android/gms/maps/model/MarkerOptions.html#alpha
(float)
        //Sets the alpha (opacity) of the marker.
        markerOptions.alpha(0.9f);
```

```
//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#addMarker(com.g
oogle.android.gms.maps.model.MarkerOptions)
        /*
        Adds a marker to this map.
        The marker's icon is rendered on the map at the location Marker.position.
        Clicking the marker centers the camera on the marker.
        If Marker.title is defined, the map shows an info box with the marker's title and snippet.
        If the marker is draggable, long-clicking and then dragging the marker moves it.
        */
        Marker marker = null;
        if (mGoogleMap!=null) {
            marker = mGoogleMap.addMarker(markerOptions);
        }

        return marker;
    }//addMarkerToMap

    //----------------------------------------------------------------------------------------------------------
--------
    public boolean changeMapType(
        int pMapType
    )
    {
        if (mGoogleMap!=null){
            switch (pMapType){
                case GoogleMap.MAP_TYPE_HYBRID:
                case GoogleMap.MAP_TYPE_NONE:
                case GoogleMap.MAP_TYPE_NORMAL:
                case GoogleMap.MAP_TYPE_SATELLITE:
                case GoogleMap.MAP_TYPE_TERRAIN:
                    mGoogleMap.setMapType(pMapType);
                    return true;
                default:
                    //unknown Google Map type
                    return false;
            }//switch
        }//if
        return false;
    }//changeMapType

    //----------------------------------------------------------------------------------------------------------
--------
    public void centerAtLocation(
        double pLatitude,
        double pLongitude,
        int pZoom
    )
    {
        //CameraUpdate camUpdate = new CameraUpdate (...); //CameraUpdate is not public

//https://developers.google.com/android/reference/com/google/android/gms/maps/CameraUpdateFactory.html

//https://developers.google.com/android/reference/com/google/android/gms/maps/CameraUpdateFactory.html#newLa
tLngZoom(com.google.android.gms.maps.model.LatLng,%20float)
        CameraUpdate someLocationAndZoomLevel = CameraUpdateFactory.newLatLngZoom(
                new LatLng(pLatitude, pLongitude),
                pZoom
        );

        /*
        public final void moveCamera (CameraUpdate update)
        Repositions the camera according to the instructions defined in the update. The move is
instantaneous, and a subsequent getCameraPosition() will reflect the new position. See CameraUpdateFactory
for a set of updates.
        Parameters : update The change that should be applied to the camera.
         */

//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#moveCamera(com.
google.android.gms.maps.CameraUpdate)
        mGoogleMap.moveCamera(someLocationAndZoomLevel);
    }//vaiParaLatLng

    //--------------------------------------------------------------------------------------------------
--------
    @Override
    public void onMapReady(
        GoogleMap pGoogleMap
    )
    {
        /*
        https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap
        int    MAP_TYPE_HYBRID    Satellite maps with a transparent layer of major streets.
        int    MAP_TYPE_NONE  No base map tiles.
        int    MAP_TYPE_NORMAL    Basic maps.
        int    MAP_TYPE_SATELLITE     Satellite maps with no labels.
        int    MAP_TYPE_TERRAIN   Terrain maps.
         */
        pGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

        mUtil.requestNecessaryPermissionsNotYetGranted(
                NECESSARY_PERMISSIONS,
                mAppRequestCode
        );

        try{

//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#setMyLocationEn
abled(boolean)
            //Enables or disables the my-location layer
            pGoogleMap.setMyLocationEnabled(true);
        }
        catch (SecurityException e){

        }

        //accessibility related

//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#setContentDescr
iption(java.lang.String)
        pGoogleMap.setContentDescription("it is a map!"); //This is used to provide a spoken description of
the map in accessibility mode. The default value is "Google Map"

        //layers related

//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#setBuildingsEna
bled(boolean)
        //Turns the 3D buildings layer on or off.
        pGoogleMap.setBuildingsEnabled(true);


//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#isIndoorEnabled
()
        /*
        Sets whether indoor maps should be enabled.
        Currently, indoor maps can only be shown on one map at a time and by default, this is the first map
added to your application.
```

```
        To enable indoor maps on another map, you must first disable indoor maps on the original map.
        If you try to enable indoor maps when it is enabled on another map, nothing will happen and this
will return false.
        When Indoor is not enabled for a map, all methods related to indoor will return null, or false.

        https://support.google.com/maps/answer/2803784?co=GENIE.Platform%3DDesktop&hl=en
         */
        pGoogleMap.setIndoorEnabled(false);


//https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap.html#setTrafficEnabl
ed(boolean)
        //Turns the traffic layer on or off.
        pGoogleMap.setTrafficEnabled(false);


//https://developers.google.com/android/reference/com/google/android/gms/maps/UiSettings.html#setCompassEnab
led(boolean)
        /*
        If enabled, it is only shown when the camera is tilted or rotated away from its default orientation
(tilt of 0 and a bearing of 0)
         */
        pGoogleMap.getUiSettings().setCompassEnabled(true); //Enables or disables the compass. => aparece no
canto superior esquerdo


//https://developers.google.com/android/reference/com/google/android/gms/maps/UiSettings.html#setMyLocationB
uttonEnabled(boolean)
        /*
        The my-location button causes the camera to move such that the user's location is in the center of
the map.
        If the button is enabled, it is only shown when the my-location layer is enabled.
         */
        pGoogleMap.getUiSettings().setMyLocationButtonEnabled(true); //Enables or disables the my-location
button => aparece no canto superior direito


//https://developers.google.com/android/reference/com/google/android/gms/maps/UiSettings.html#setZoomControl
sEnabled(boolean)
        /*
        If enabled, the zoom controls are a pair of buttons (one for zooming in, one for zooming out) that
appear on the screen.
         */
        pGoogleMap.getUiSettings().setZoomControlsEnabled(true); //aparece no canto inferior direito


//https://developers.google.com/android/reference/com/google/android/gms/maps/UiSettings.html#setMapToolbarE
nabled(boolean)
        /*
        If enabled, and the Map Toolbar can be shown in the current context, users will see a bar with
various context-dependent actions,
        including 'open this map in the Google Maps app' and 'find directions to the highlighted marker in
the Google Maps app'.
         */
        //pGoogleMap.getUiSettings().setMapToolbarEnabled(true); //não consigo visualizar no emulador

        mGoogleMap = pGoogleMap;
    }//onMapReady
}//AmGoogleMap
```

Using an example AmGoogleMap to see a map in some Activity

In the (main?)Activity where a map is needed, declare the following data members that will allow the user the change the map type, as suggested in the picture.



```
public final static int CALL_ME_ON_THIS_CODE_WHEN_THE_GOOGLE_MAP_IS_READY_FOR_DISPLAY = 123;
    public final static String MAP_HYBRID = "HYBRID";
    public final static String MAP_NONE = "NONE";
    public final static String MAP_NORMAL = "NORMAL";
    public final static String MAP_SATELLITE = "SATELLITE";
    public final static String MAP_TERRAIN = "TERRAIN";
    public final String[] MAP_TYPES =
    {
        MAP_HYBRID,
        MAP_NONE,
        MAP_NORMAL,
        MAP_SATELLITE,
        MAP_TERRAIN
    };

    MapFragment mMapFragment;
    AmGoogleMap mAmGoogleMap;
    AmUtil mUtil;
    Spinner mSpnMapType;

    AdapterView.OnItemSelectedListener mItemSelectedHandler = new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            String strSelectedOption = parent.getItemAtPosition(position).toString();

            int iGoogleMapType = 0;
            switch(strSelectedOption){
                case MAP_HYBRID:
                    iGoogleMapType = GoogleMap.MAP_TYPE_HYBRID;
                    break;
                case MAP_NONE:
                    iGoogleMapType = GoogleMap.MAP_TYPE_NONE;
                    break;
                case MAP_NORMAL:
                    iGoogleMapType = GoogleMap.MAP_TYPE_NORMAL;
                    break;
                case MAP_SATELLITE:
                    iGoogleMapType = GoogleMap.MAP_TYPE_SATELLITE;
                    break;
                case MAP_TERRAIN:
                    iGoogleMapType = GoogleMap.MAP_TYPE_TERRAIN;
                    break;
            }//switch

            if (mAmGoogleMap !=null){
                mAmGoogleMap.mudarTipoDeMapa(iGoogleMapType);

                mAmGoogleMap.criaMarcadorNoMapa(new LatLng(11.1, 11.1), "Bla!");
                mAmGoogleMap.centrarEmGeoPosicao(11.1, 11.1, 10);
            }//if
       }//mItemSelectedHandler

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    };
```

Following the "init" pattern, add something like

```
void init(){
        mUtil = new AmUtil(this);

        mSpnMapType = findViewById(R.id.idSpnMapType);
        mSpnMapType.setOnItemSelectedListener(mItemSelectedHandler);

        mUtil.populateSpinnerWithOptions
        (
            mSpnMapType,
            MAP_TYPES
        );

        mAmGoogleMap = new AmGoogleMap(
            this,
            CALL_ME_ON_THIS_CODE_WHEN_THE_GOOGLE_MAP_IS_READY_FOR_DISPLAY
        );

        mMapFragment = (MapFragment) getFragmentManager().findFragmentById(R.id.idGMapFragment);
        mMapFragment.getMapAsync(mAmGoogleMap);
    }//init
```

The app should run and allow map type selection, nothing more.