ATLS 4120/5120: Mobile Application Development
Week 14: Google Maps

Google Maps
https://developers.google.com/maps/
Google Maps can be used on any platform through their API. (Android | API | Guides); Map objects

Google Play Services
https://developers.google.com/android/guides/overview#the_google_play_services_apk
Google Play Services are distributed as an APK(Android Package Kit) from the Google Play Store. You must have these installed on your device or emulator to run an app using Google Maps.
The nice thing is that users can update these any time from the Google Play store, they don't have to wait for the OEM or carrier to send out an update.

Android and Google Maps
Maps Android API Getting Started https://developers.google.com/maps/documentation/android-api/start

Map
Create a new Android Studio project called map
Phone and Tablet min SDK API 19 (please chose 21 or lower or I won't be able to test your app)
Make sure to choose the API version of your device or lower (Settings | More | About)
Google Maps Activity
Activity Name: MapsActivity
Layout Name: activity_maps
Title: Map
No hierarchical parent needed

Google Play Services
Tools | Android | SDK manager
SDK Tools tab
Check Google Play Services to install

Files
activity_maps.xml is the layout for the map. Notice it contains only a fragment element and that has the id "map".
Fragments allow you to break your activities up into smaller modular components which can easily be reused and adapted for different device sizes, orientation, or other criteria.
You can have one or more fragments embedded in an activity.
For this app we're just going to leave the one fragment created for us.
The Fragment class has many of the same methods as the Activity class.

MapsActivity.java
Notice all the imports at the top in order to use maps.
Our class MapsActivity extends FragmentActivity since it's using fragments.
It also implements the OnMapReadyCallback interface.
In Java an interface is very similar to a protocol in Swift, so when you implement an interface it's like adopting a protocol and now you have access to its methods.
The public method in the OnMapReadyCallback interface is onMapReady() which is called when a map is ready to be used.

https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap
A GoogleMap instance is sent to onMapReady() and that is assigned to the mMap instance.
A LatLng object is defined with the coordinates for Sydney, Australia
A marker is added to those coordinates.
https://developers.google.com/maps/documentation/android-api/views#the_camera_position
The map view is modeled as a camera looking down on a flat plane. The position of the camera (and hence the rendering of the map) is specified by the following properties

- target (latitude/longitude location)
- bearing (orientation)
- tilt (viewing angle)
- zoom

moveCamera() moves the map to the location coordinates.

google_maps_api.xml contains instructions on getting a Google Maps API key before you try to run the application.

Google Maps API key
In order to use Google Maps from any form of application (not just Android ones), you need to source and use a Google API key and configure your Google account for the Maps API.

Copy and paste the link in the google_maps_api.xml file into a browser.

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=07:8D:07:3C:C1:53:34:E7:F6:11:2F:96:91:AB:6A:72:CC:94:5E:3D%3Bcom.example.aileen.map
Create a project (takes a few minutes)
Create API key
Copy your API key
https://console.developers.google.com/apis/credentials?project=driven-photon-150723

In the google_maps_api.xml file replace "YOUR_KEY_HERE" with your API key (no quotes needed)

The AndroidManifest.xml file has two entries that we haven't seen before.

<**uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"** />
Allows Android to access location using on-device sensors and radios that can approximate location at a course level. This includes cellular (tower) data, wi-fi signals, and GPS.
There are other permissions such as **"android.permission.ACCESS_COARSE_LOCATION"** which does not use GPS.

```
<meta-data
   android:name="com.google.android.geo.API_KEY"
   android:value="@string/google_maps_key" />
```
Gives your app access to the Google Maps API using your API key

Run the app on your device. You should see the marker on Sydney, Australia.
(ignore Instant Run errors)
(update Google Play services on your device if needed)

You might get this error on older phones:
Error:Execution failed for task ':app:transformClassesWithDexForDebug'.

On your device go into Settings | Apps | Google Play Services OR
Settings | More | Application Manager | All (Note: You might need to scroll to the right to see this) | Google Play Services.
Update if available. Make sure it's enabled. Get the version number.
In your app go into Gradle Scripts and open build.gradle(Module: app) and under dependencies change this line to the version of Google Play Services on your device(only use 1 significant digit, so not .01):
implementation **'com.google.android.gms:play-services-maps:11.6.0'**
(ex: change to implementation **'com.google.android.gms:play-services-maps:8.3.0'**)

Sync after the change and try to run it on your device.