ATLS 4120: Mobile Application Development
Week 15: Google Maps

Android and Google Maps
Google Maps can be used on any platform through their API.
Maps Android API Getting Started https://developers.google.com/maps/documentation/android-api/start

Google Play Services
https://developers.google.com/android/guides/overview
Google Play Services is a client library that contains the interfaces to the individual Google services and allows you to obtain authorization from users to gain access to these services with their credentials. It also contains APIs that allow you to resolve any issues at runtime, such as a missing, disabled, or out-of-date Google Play services APK.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.

Map
Create a new Android Studio project called map
Phone and Tablet min SDK API 21 (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
Name: Map
Language: Java
Minimum API level: 21 (or the level of your device if it's lower)

Google Play Services
Tools | 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 FragmentActivity class has many of the same methods as the AppCompatActivity 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.
A GoogleMap called mMap is defined.
https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap

The class also implements the OnMapReadyCallback interface.

In Java an interface is an abstract class so when you implement an interface you have access to its methods.
It's very similar to adopting a protocol in Swift.
The public method in the OnMapReadyCallback interface is onMapReady() which is called when a map is ready to be used.
In onCreate() we get access to the map fragment using the map id.
The getMapAsync() method sets a callback object which will be triggered when the GoogleMap instance is ready to be used.

- If google play services is not installed the callback won't be triggered
- The parameter is "this" because it's the current class that implements the OnMapReadyCallback interface and the onMapReady() method

onMapReady() is called once the map is ready to be used

- A GoogleMap instance associated with the MapFragment is automatically sent to onMapReady() and then assigned to the mMap instance.
- A LatLng object is defined with the coordinates for Sydney, Australia https://developers.google.com/android/reference/com/google/android/gms/maps/model/LatLng
- 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.

res/values/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)

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 and returns a location with an accuracy approximately equivalent to a city block.

Android 10 (API level 29) or higher includes the `ACCESS_BACKGROUND_LOCATION` permission if you need to access the device location while your app is in the background, although this is not recommended. On Android 8.0 (API level 26) and higher, if an app is running in the background when it requests the current location, the device calculates the location only a few times each hour.

```
<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

The template also adds a Google Play Services dependency to build.gradle(Module: app). This dependency exposes the Google Maps and Location Services APIs to the application.

```
implementation 'com.google.android.gms:play-services-maps:16.1.0'
```

Run the app on your device. You should see the marker on Sydney, Australia.

(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:16.0.0'
(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.

To run the app in the emulator Google Play Services must be installed. This can be a lengthy process but it ran on the Pixel emulator with no additional steps needed.