|  |
| --- |
| Locating a Place |

***Resources and References***

**Google Map for Android** <https://developers.google.com/maps/documentation/android-api/>

**Google API Console** <https://console.developers.google.com/apis>

**Coordinates** <http://modernsurvivalblog.com/survival-skills/basic-map-reading-latitude-longitude/>

Table of Contents

[Locating a Place 1](#_Toc517860262)

[Section A: Intro to Google Maps 1](#_Toc517860263)

[Section B: Preparation for Google Maps 2](#_Toc517860264)

[Section C: Configuring / Building Android Project 7](#_Toc517860265)

[Section D: Modifying the Google Map 10](#_Toc517860266)

[Section E: Controlling the map 11](#_Toc517860267)

[Section F: Marking locations on a Map 14](#_Toc517860268)

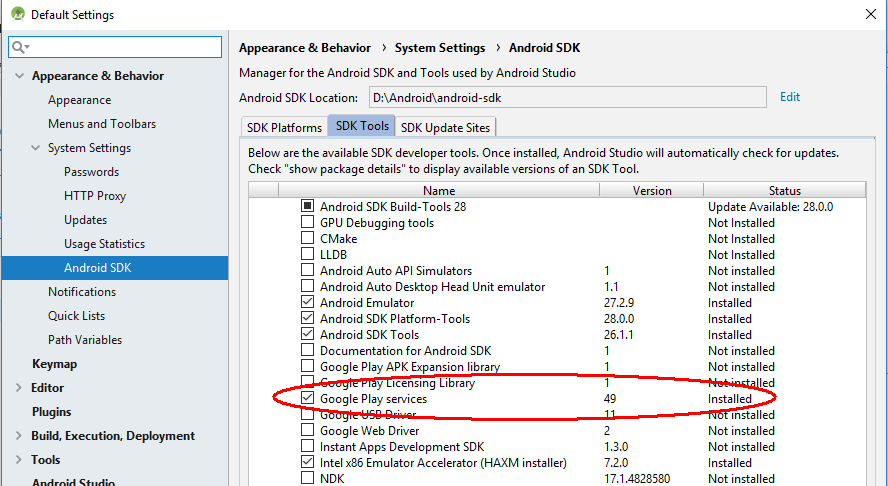
[Handling the Problem Statement 15](#_Toc517860269)

## Section A: Intro to Google Maps

1. Most, if not all, Android phones come with a default Google Maps application. You can also create your own customized map application with features that are not provided in the default Maps application in your own app.
2. The current Google Map implementation in Android is known as Google Maps Android API v2. It is not part of the open source Android OS and provided as a separate library by Google. Due to this, there are some additional steps to be performed to implement a Google Map component in Android app. These are the following tasks to be performed:
   1. Install the Google Play services SDK
   2. Google Maps restricted access to it through the use of API key. To request for one, you need to provide the SHA1 hash value of the debug certificate of your installed Android SDK.
   3. API request needs to be done via logging into the Google API Console using a Google Account. You will need to register a project in the Google APIs Console, to request for a Google Maps API Key.
   4. Specify settings in the Android app Manifest file.
   5. Add a map fragment to an Activity.
   6. Run your app!
   7. To publish the app to the Google Play, you will need to obtain a new Google Maps API Key from the certificate used to sign the app.

## Section B: Preparation for Google Maps

1. The first step is to make sure that Google Play services is installed. You could perform an installation or perform a check at the SDK Manager.



1. Create a new Android Project to kick start this. Create the following project:

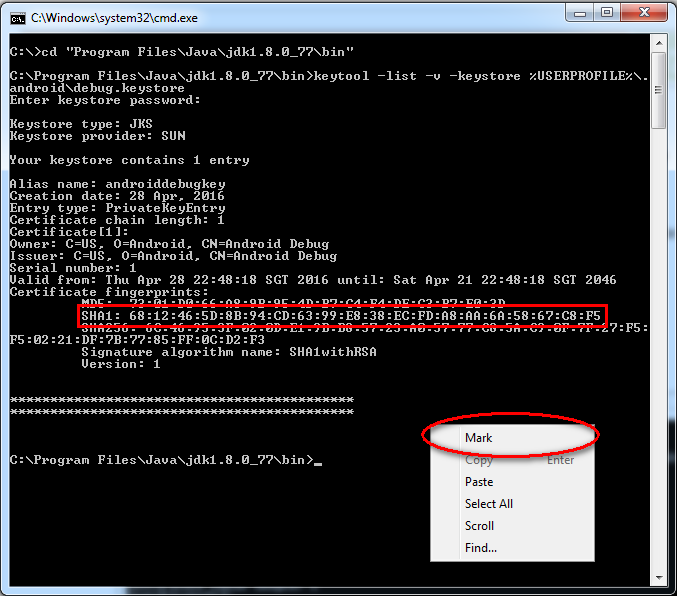
|  |  |
| --- | --- |
| Project Name | Demo Map |
| Package | com.myapplicationdev.android.demomap; |
| Activity Name | MainActivity |
| Layout Name | activity\_main.xml |
| Min SDK | API 16 |

\*Import into Github

1. To obtain a Google Maps API Key, you need to provide a SHA1 hash digest of a certificate used to sign an Android app. Every Android app needs to be signed with a certificate to be installed on an Android phone.
2. Android Studio IDE automate this signing and this has been transparent to the developer. The Android SDK maintained a keystore containing a **debug certificate** for this purpose. We will use this certificate to request for a Google Map API key.
3. Go to the **Command Prompt**.
4. Change directory to your **Java\bin** folder.
5. Execute the command

**keytool -list -v -keystore %USERPROFILE%\.android\debug.keystore**

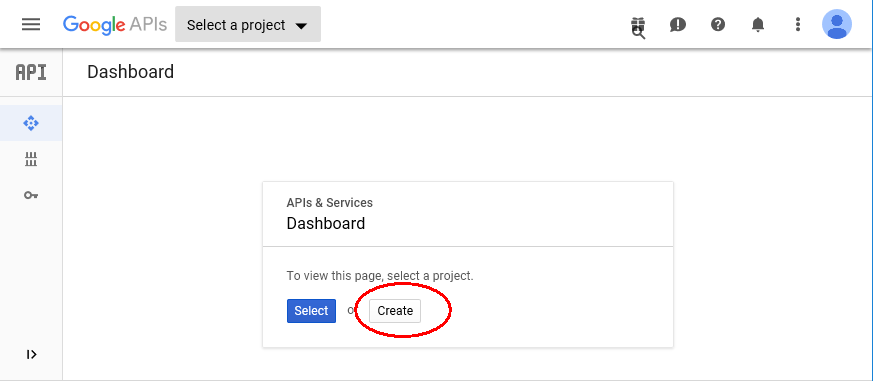
The password to the keystore is “**android**”. *The password won’t show while typing. Just type it.*



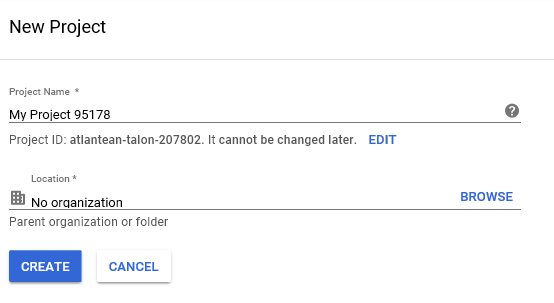
1. Right click in the common prompt window and select **Mark.**
2. Highlight the SHA1 value and hit **Enter** to copy.
3. Proceed to sign up for Google API at <https://console.developers.google.com/projectselector/apis/dashboard>

Login with your Gmail account.

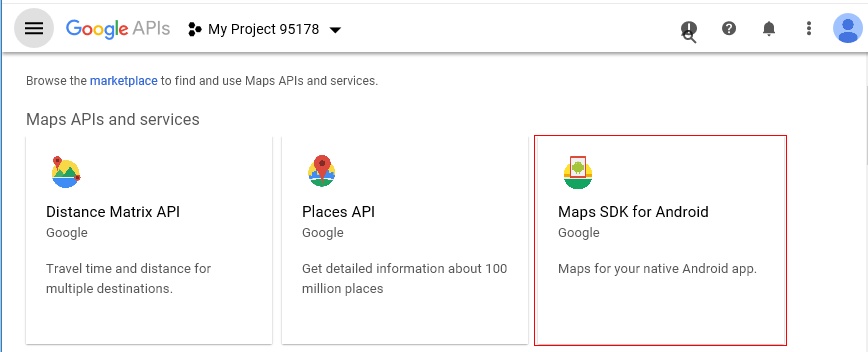
1. API Keys are tied to projects. You need to create a new project.



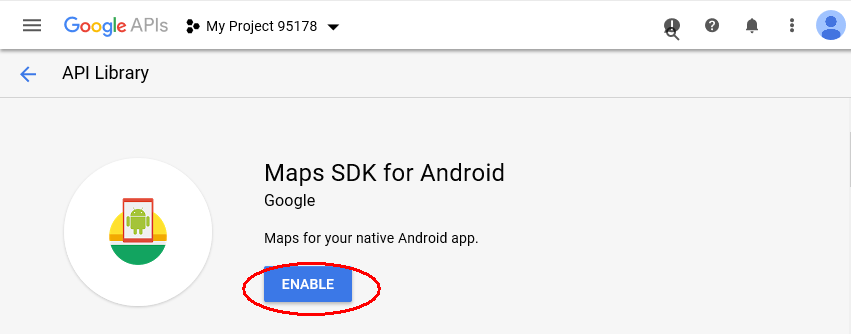
1. Enter the project name as **C347 Project** and click on the **Create** button.



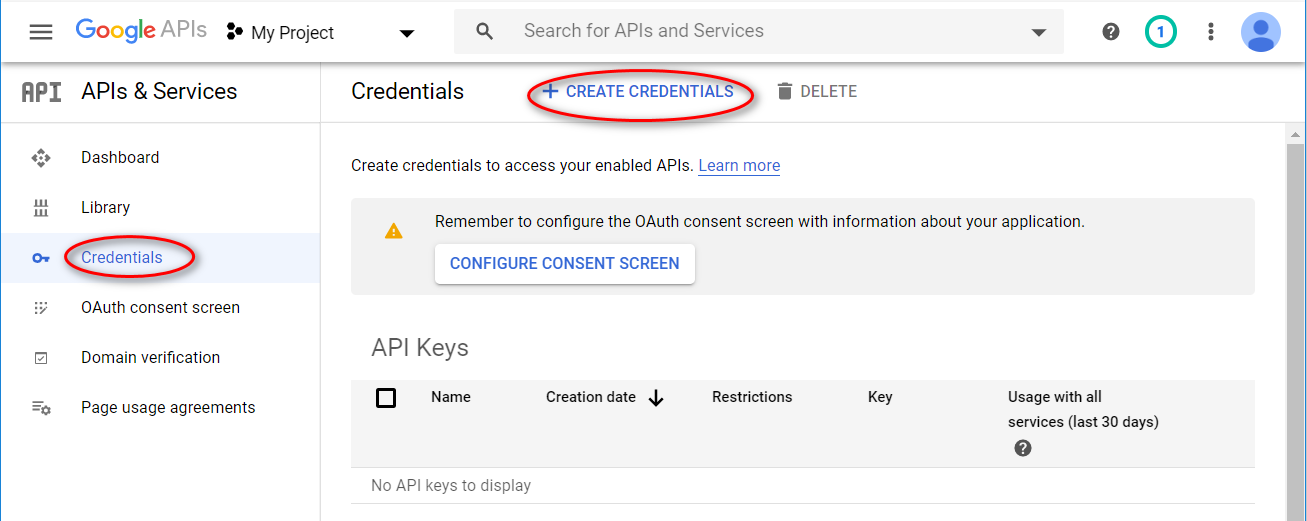
1. After the project creation, look for Google Maps API under the “Library” tab.



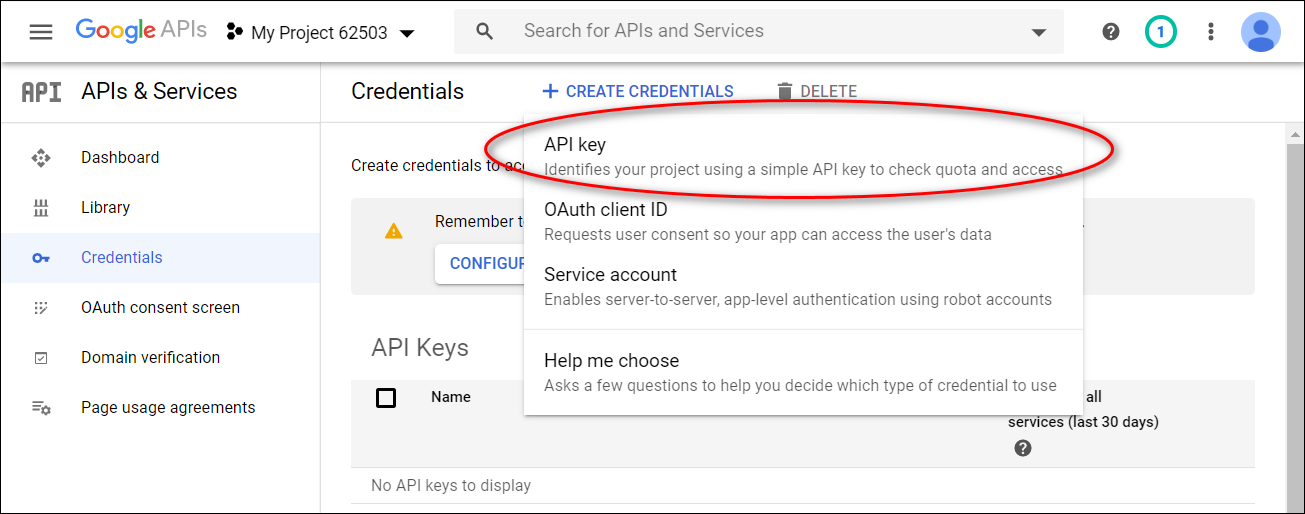
1. Click on the **Enable** button for enabling Google Maps Android API.



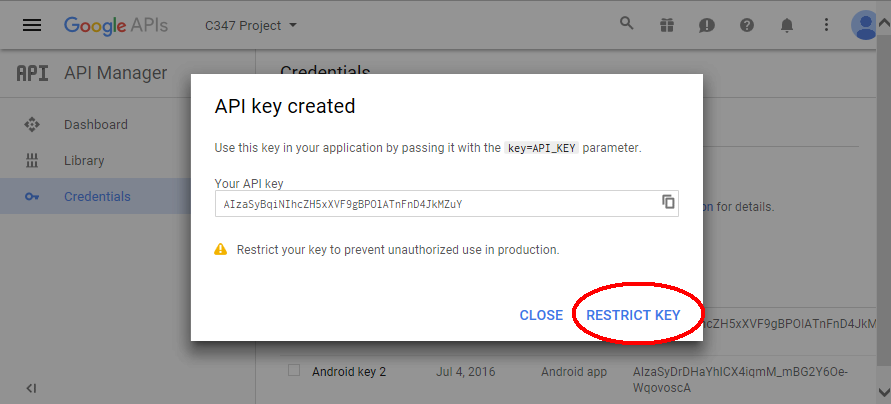
1. Click on **Credentials** button and subsequently “Create credentials” button.



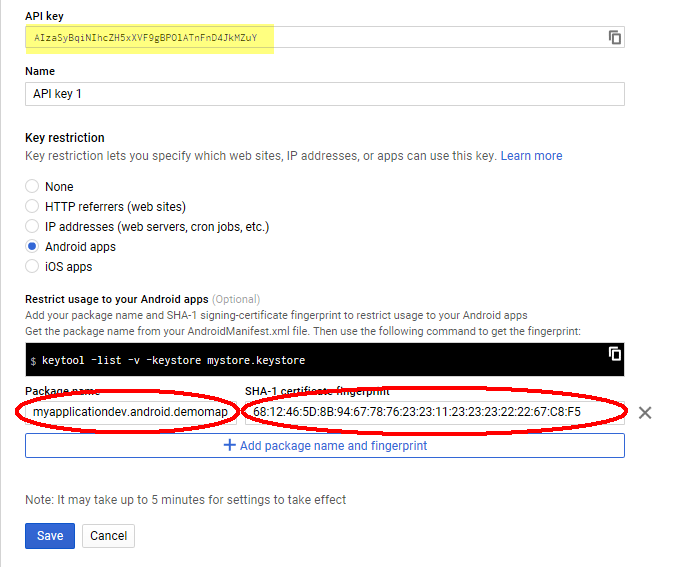
1. Choose API Key as follows.

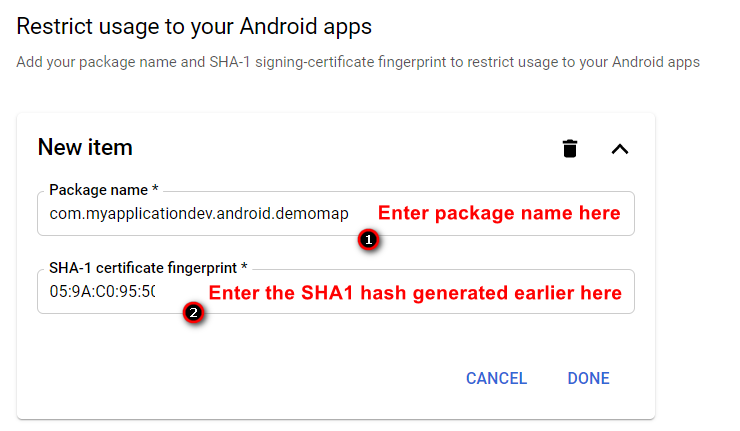


1. The API Key is created. Subsequently, click on restrict key.

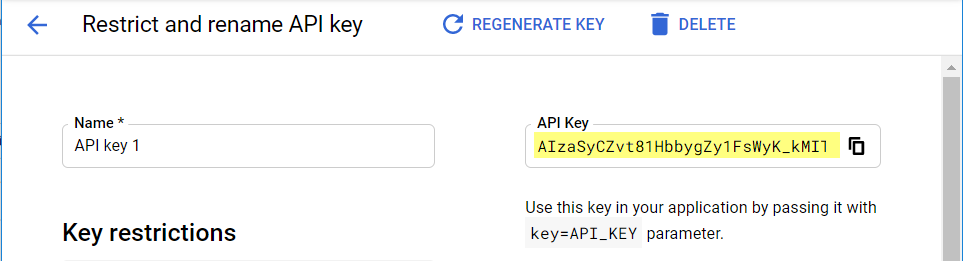


1. Input the package name of the Android app as well as the SHA1 digest obtained earlier.





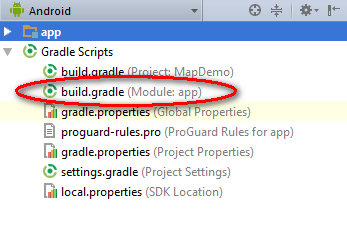
1. Copy the generated API Key for- use later on.



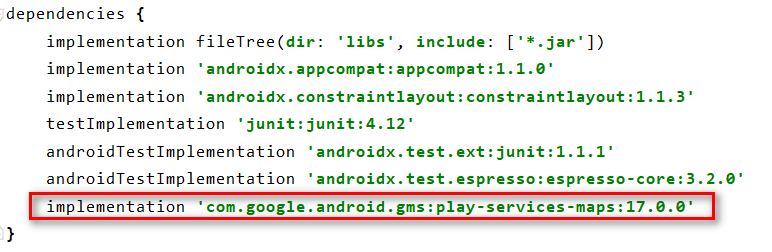
## Section C: Configuring / Building Android Project

1. You need to configure Gradle (the builder of Android Studio IDE) to include the necessary libraries to implement Google Maps.

Open the file **build.gradle** (Module: app) under the Grade Scripts

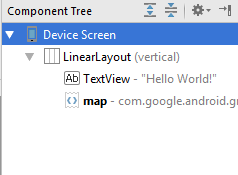


Add the following code in the red box. This will enable the project to use Goole Play Services. Gradle will **need** Internet access to download the necessary libraries.



Do remember to perform a Gradle sync. *Click* ***sync now*** *on top.*

1. At the **activity\_main.xml**, add the following component:



1. Permissions are needed for Google Maps API v2. We need to state the required permissions in the AndroidManifest.xml

Add the following permissions:

* **android.permission.WRITE\_EXTERNAL\_STORAGE** - Allows the API to cache map tile data in the device's external storage area.
* **android.permission.ACCESS\_FINE\_LOCATION** - Allows the API to use the Global Positioning System (GPS) to determine the device's location to within a very small area.
* **android.permission.ACCESS\_COARSE\_LOCATION** - Allows the API to use Wi-Fi or mobile cell data (or both) to determine the device's location.
* **android.permission.ACCESS\_LOCATION\_EXTRA\_COMMANDS** - Allows an application to access extra location provider commands
* **android.permission.ACCESS\_NETWORK\_STATE** - Allows the API to check the connection status in order to determine whether data can be downloaded.
* [**android.permission.INTERNET**](http://developer.android.com/reference/android/Manifest.permission.html#INTERNET)- Used by the API to download map tiles from Google Maps servers.

*\*Note: Some of these permissions are not needed to be stated for new API level. But for backward compatibility, we would leave them in the AndroidManifest.xml*

1. Your AndroidManifest.xml should have the following:



1. Add your API key generated earlier into the AndroidManifest.xml as shown below. It will be used to authenticate the access to the Google Maps services.

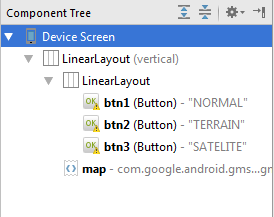


1. Run the app on a physical Android phone. You should see a map shown on the Activity.

*\*Note: For Emulator, only API 27 and above with Google Play should work.*

## Section D: Modifying the Google Map

1. Modify the UI to the following content in **activity\_main.xml**

1. At the **MainActivity.java**, append the following code:



 The code at line 14-19 mean to get the reference to the Google Map object. The object will only be ready when it is fully loaded. The time this object needs depends on the Internet connection, thus it’s tracked separately.

mapFragment.getMapAsync(new OnMapReadyCallback(){  
 @Override  
 public void onMapReady(GoogleMap googleMap) {  
 map = googleMap;  
 }  
 });

When the object is initialized and ready, the code in onMapReady() will be run, thus obtaining a reference to the global variable to be used later in the Activity.

Run the app and the map type will be changed when any button is clicked.

## Section E: Controlling the map

1. Location can be expressed in coordinates. Coordinates consist of Latitude and Longitude.
2. A simplified explanation of latitude and longitude can be found here: <http://modernsurvivalblog.com/survival-skills/basic-map-reading-latitude-longitude/>
3. You can easily find the coordinate of a location here:  
   <http://www.latlong.net/>
4. We will modify the Demo Map app to zoom to Causeway Point, Woodlands.
5. Back to the **MainActivity.java**, append the following code:
6. Run the app. The map will be zoomed in to Causeway Point.



1. Some of the common methods used to move the map are as follow:
   1. **CameraUpdateFactory.zoomIn()** – Provides a CameraUpdate instance zoomed in by one level.
   2. **CameraUpdateFactory.zoomOut()** - Provides a CameraUpdate instance zoomed out by one level.
   3. **CameraUpdateFactory.zoomTo(float)** - Generates a CameraUpdate instance that changes the zoom level to the specified value.
   4. **CameraUpdateFactory.zoomBy(float)** – Provides a CameraUpdate instance with a zoom level increased or decreased by the specified amount.
   5. **CameraUpdateFactory.newLatLng(LatLng)** - Creates a CameraUpdate instance that changes the camera's target latitude and longitude.
   6. **CameraUpdateFactory.newLatLngZoom(LatLng, float)** - Generates a CameraUpdate instance that changes the camera's latitude, longitude and zoom.

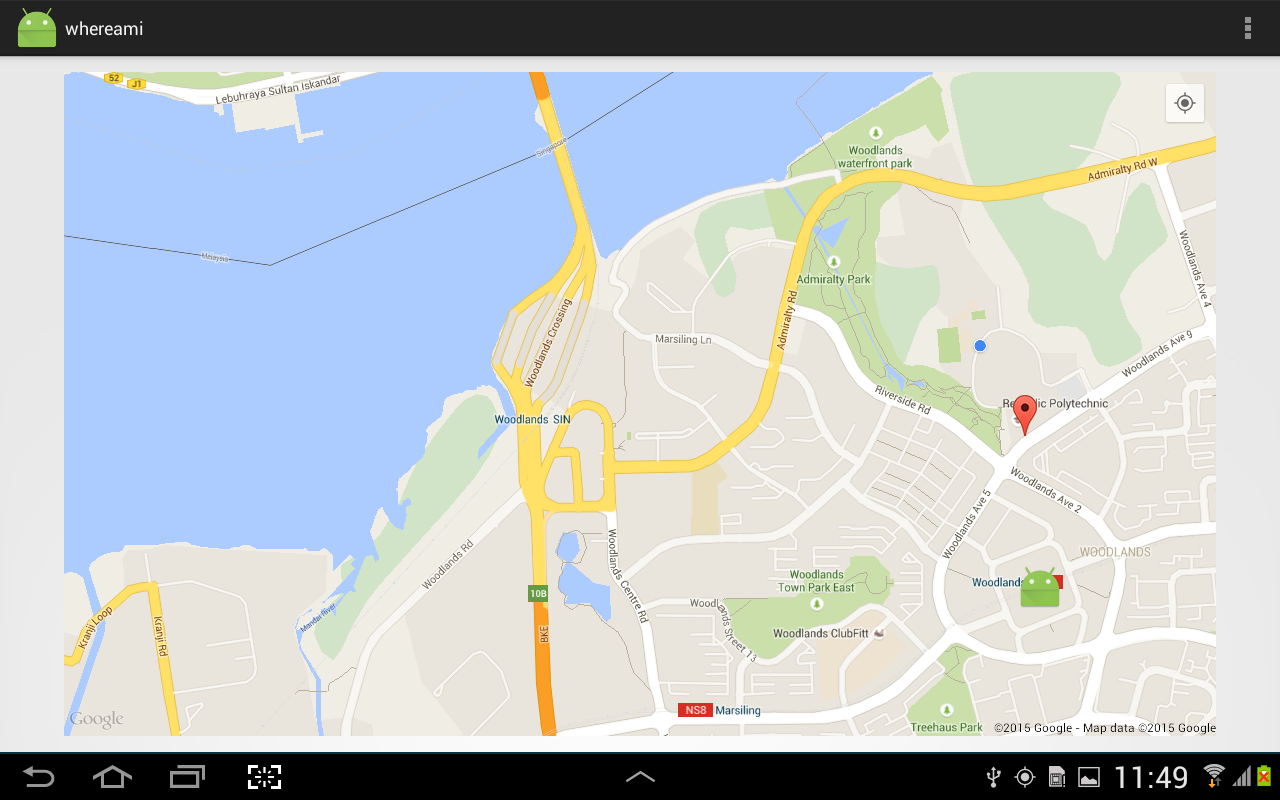
Do experiment with the methods above.

1. Now that you are able to move around the map. You could also set some controls on the map by calling some methods. Note that the **compass icon will appear only if you rotate the map to not align to north**.

|  |  |  |
| --- | --- | --- |
| a) | Compass | UiSettings ui = map.getUiSettings();  ui.setCompassEnabled(true); |
| b) | Zoom controls | UiSettings ui = map.getUiSettings();  ui.setZoomControlsEnabled(true); |

## Section F: Marking locations on a Map

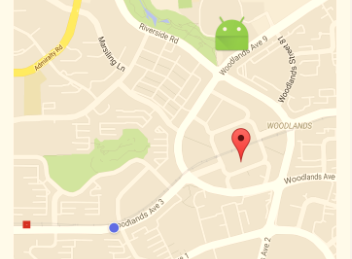
1. You can show your current location on the Map.



This can be done by inserting the code below. The permission check is especially important for API 23 (Marshmallow) onwards. This is because a user can choose to revoke a permission after the installation of the app. As such, an in-code permission check is necessary to prevent run time exception.

*\*Note: Implement the onRequestPermissionsResult() method as we did before to check and mark/enable the current location if the permission has been granted by user. You may need to refresh the app to see the current location.*

1. You could also place some markers on the map. The followings are 2 markers and the phone’s current location.



The phone’s location

Causeway Point

RP

1. To place a marker, you need to define the marker’s coordinates first. Below are 2 examples of markers, shown in the image.



*\*Note: Don’t forget to place an image in drawable (line 16)*

1. Run the app. When you tap on the marker, some info will be shown.



## Handling the Problem Statement

What are the necessary preparation to enable Google Maps on Android app?

1. What are the permission?
2. What is the UI to display the Google Map
3. What are the controls needed for the map?
4. How many markers are needed?
   1. What are the coordinates?
   2. What are the graphical representation?
   3. Are there any info associated with the marker?