Mobile Programming

Dr. Nader Mahmoud

Lecturer at Computer Science department

Course Content

- Chapter 1: Getting Started with Android Programming
- Chapter 2: Using Android Studio for Android Development
- Chapter 3: Activities, Fragments, and Intents
- Chapter 4: Getting to know the Android User Interface
- Chapter 5: Designing Your User Interface with Views
- Chapter 6: Displaying Pictures and Menus with Views
- Chapter 7: Data Persistence
- Chapter 8: Content Providers
- Chapter 9: Messaging
- Chapter 10: Location-Based Services
- Chapter 11: Networking
- Chapter 12: Developing Android Services

Agenda

- **Chapter 10 - Location Based Services**
  - Displaying Maps
    - Creating the project
    - Obtaining Maps API key
    - Displaying Map
    - Displaying Zoom control
    - Changing views
    - Navigating to a specific location
    - Getting Location Information
    - Geocoding and Reverse Geocoding
  - Getting Location Data

Architecture of Android

allow applications to obtain periodic updates of the device's geographical location

APPLICATIONS

- Home
- Contacts
- Phone
- Browser
- ...

APPLICATION FRAMEWORK

- Activity Manager
- Window Manager
- Content Providers
- View System
- Package Manager
- Telephony Manager
- Resource Manager
- Location Manager
- Notification Manager

LIBRARIES

- Surface Manager
- Media Framework
- SQLite
- OpenGL/ES
- FreeType
- WebKit
- SGL
- SSL
- libc

ANDROID RUNTIME

- Core Libraries
- Dalvik Virtual Machine

LINUX KERNEL

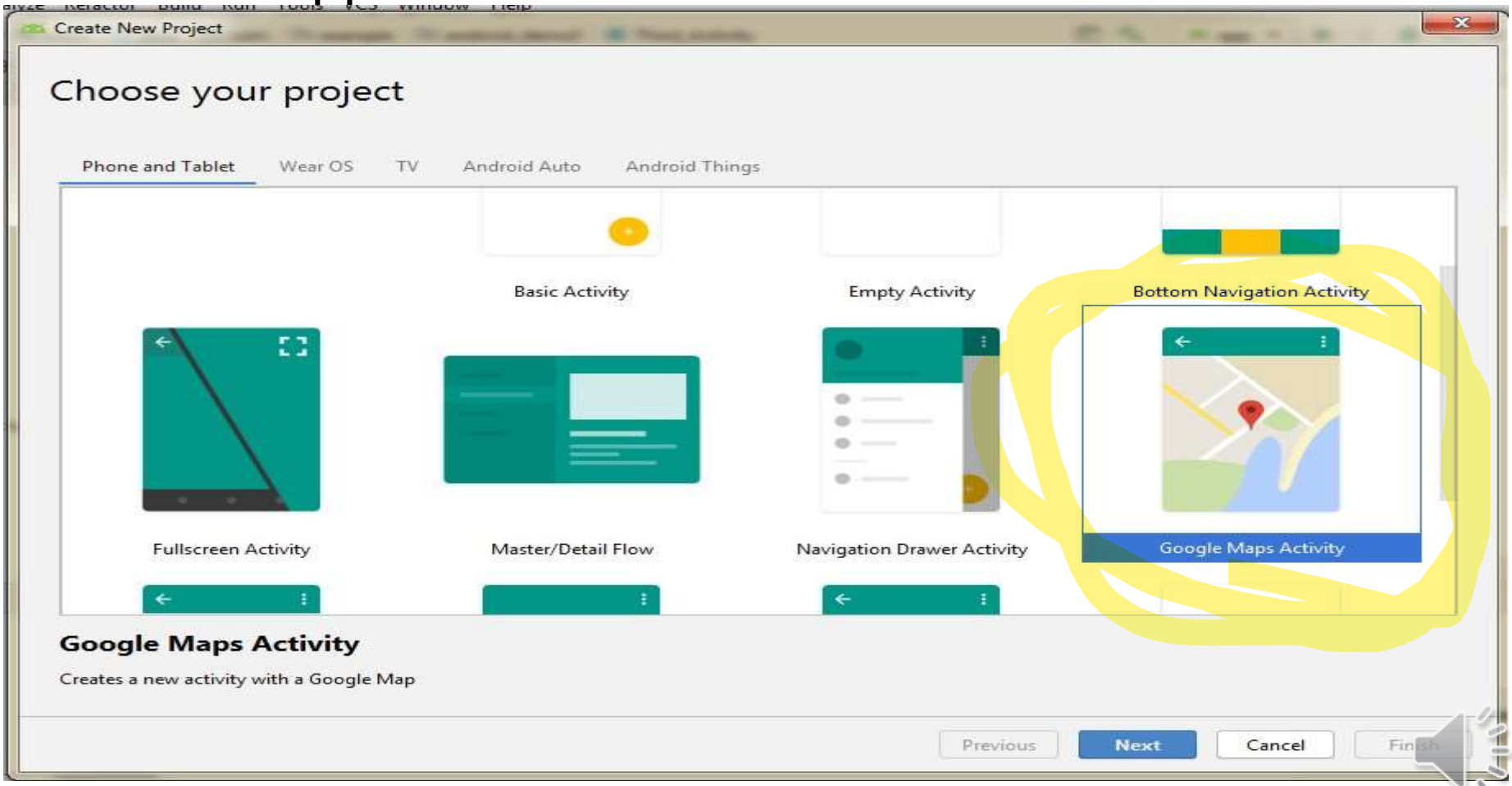
- Display Driver
- Camera Driver
- Flash Memory Driver
- Binder (IPC) Driver
- Keypad Driver
- Wi-Fi Driver
- Audio Drivers
- Power Management

Displaying Maps

- Google Maps is one of the many applications bundled with the Android
- In addition to simply using the Maps application, you can also embed it into your own applications
- This section describes how to use Google Maps in your Android applications and programmatically perform the following:
  - Change the views of Google Maps.
  - Obtain the latitude and longitude of locations in Google Maps.
  - Perform geocoding and reverse geocoding (translating an address to latitude and longitude and vice versa).

Displaying the Map

- You are now ready to display Google Maps in your Android application



Obtaining the Maps API Key

- Beginning with the Android SDK release v1.0, you need to apply for a free Google Maps API key before you can integrate Google Maps into your Android application

- To get a Google Maps key, open the google_maps_api.xml file that was created in your LBS project.

- Within this file is a link to create a new Google Maps key. Simply copy and paste the link into your browser and follow the instructions.

Obtaining the Maps API Key

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MapsAndroidDemo > app > src > debug > res > values > google_maps_api.xml

Android

app
- manifests
- java
- res
  - drawable
  - layout
  - mipmap
  - values
    - colors.xml
    - google_maps_api.xml (debug)
    - strings.xml
    - styles.xml
- Gradle Scripts

MapsActivity.java | google_maps_api.xml

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType

    You can also add your credentials to an existing key, using these values:

    Package name:
    46:F1:68:24:B7:4A:16:35:D0:EC:D4:31:6D:EC:56:10:ED:EF:31:EA

    SHA-1 certificate fingerprint:
    46:F1:68:24:B7:4A:16:35:D0:EC:D4:31:6D:EC:56:10:ED:EF:31:EA
```

resources

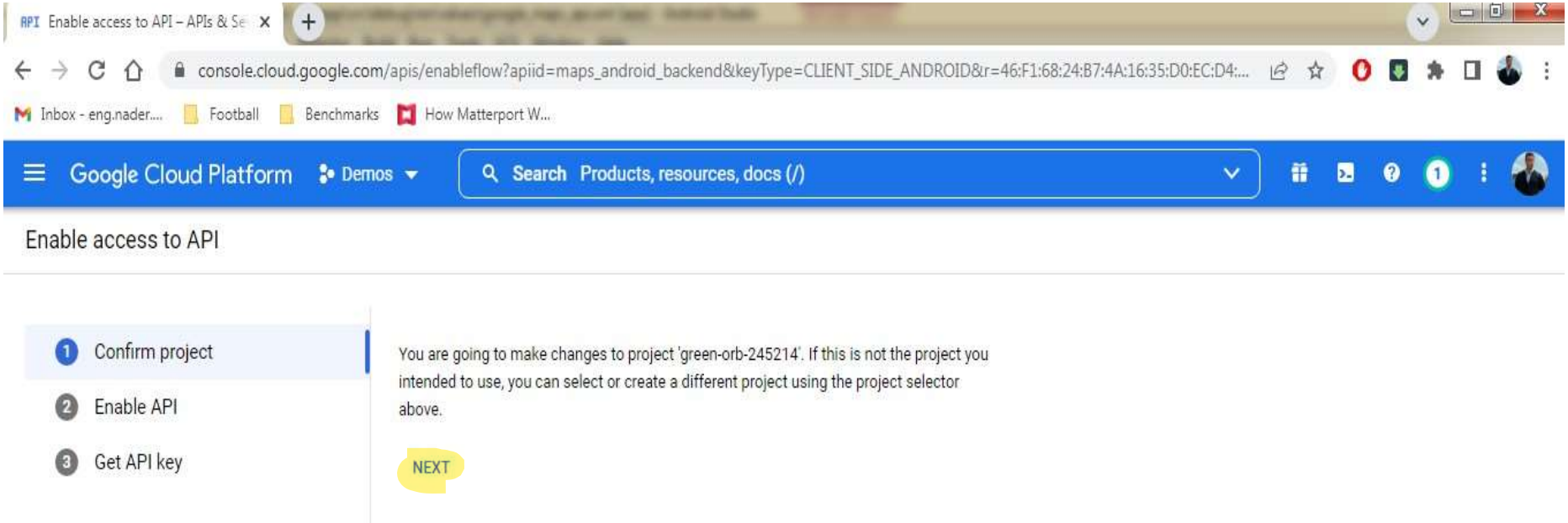
Build: Build Output | Sync

Build: build failed at 5/4/2022 5:34 PM	1 m 37 s 3 ms
Run build E:\MapsAndroidDemo	1 m 35 s 600 ms
Load build	10 ms
Evaluate settings	9 ms
Finalize build cache configuration	
Configure build	1 s 597 ms
Calculate task graph	1 m 33 s 737 ms

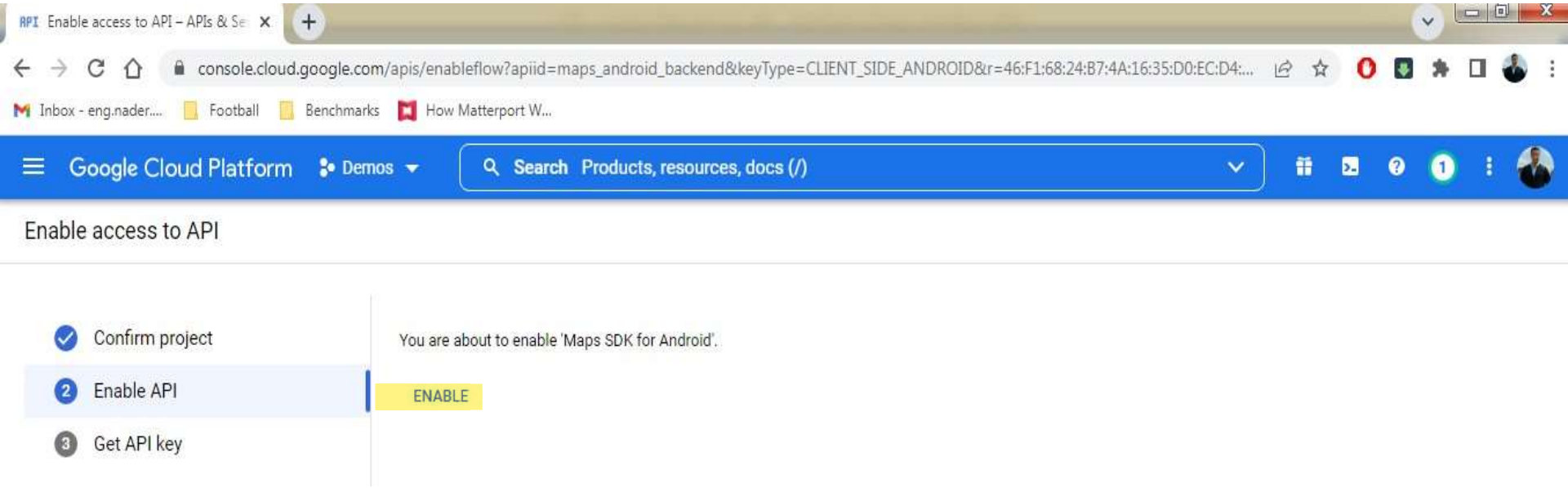
Read timed out

TODO | Terminal | Build | 6: Logcat | Event Log

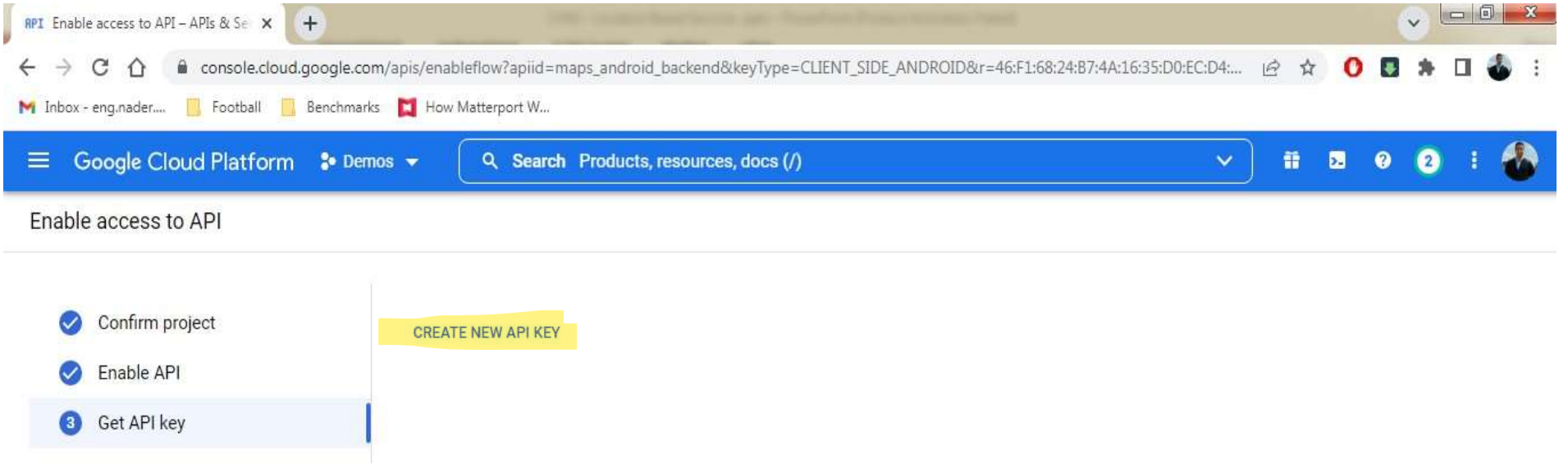
Obtaining the Maps API Key



Obtaining the Maps API Key



Obtaining the Maps API Key



Obtaining the Maps API Key

console.cloud.google.com/apis/enableflow?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=46:F1:68:24:B7:4A:16:35:D0:E...

Inbox - eng.nader... | Football | Benchmarks | How Matterport W...

Google Cloud Platform | Demos | Search Products, resources, docs (/)

Enable access to API

- Confirm project
- Enable API
- Get API key

Copy your new API key below. This is always available on the credentials page.

API key 2

AIzaSyCQZ4l9sgoHIi-yQHAdZfNnvHyzzYmGWhg

Obtaining the Maps API Key

- Replace the "google_maps_key“ string in this file.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MapsAndroidDemo 〉 app 〉 src 〉 debug 〉 res 〉 values 〉 google_maps_api.xml

app

Android

- app
  - manifests
  - java
  - res
    - drawable
    - layout
    - mipmap
    - values
      - colors.xml
      - google_maps_api.xml (debug)
      - strings.xml
      - styles.xml
- Gradle Scripts

MapsActivity.java | google_maps_api.xml

```
13
14      SHA-1 certificate fingerprint:
15      46:F1:68:24:B7:4A:16:35:D0:EC:D4:31:6D:EC:56:10:ED:EF:31:EA
16
17      Alternatively, follow the directions here:
18      https://developers.google.com/maps/documentation/android/start#get-key
19
20      Once you have your key (it starts with "AIza"), replace the "google_maps_key"
21      string in this file.
22      -->
23      <string name="google_maps_key" templateMergeStrategy="preserve"
24          translatable="false">YOUR_KEY_HERE</string>
25  </resources>
26
```

resources

Build: Build Output | Sync

- Build: build failed at 5/4/2022 5:34 PM — 1 m 37 s 3 ms
  - Run build E:\MapsAndroidDemo — 1 m 35 s 600 ms
    - Load build — 10 ms
      - Evaluate settings — 9 ms
        - Finalize build cache configuration
    - Configure build — 1 s 597 ms
    - Calculate task graph — 1 m 33 s 737 ms

Read timed out

TODO | Terminal | Build | 6: Logcat | Event Log

Gradle build finished in 1 m 37 s 80 ms (13 minutes ago)

21:25 CRLF UTF-8 4 spaces

Obtaining the Maps API Key

- Replace the "google_maps_key“ string in this file.

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

MapsAndroidDemo 〉 app 〉 src 〉 debug 〉 res 〉 values 〉 google_maps_api.xml

app

Android

- app
  - manifests
  - java
  - res
    - drawable
    - layout
    - mipmap
    - values
      - colors.xml
      - google_maps_api.xml (debug)
      - strings.xml
      - styles.xml
- Gradle Scripts

MapsActivity.java | google_maps_api.xml

```
13
14      SHA-1 certificate fingerprint:
15      46:F1:68:24:B7:4A:16:35:D0:EC:D4:31:6D:EC:56:10:ED:EF:31:EA
16
17      Alternatively, follow the directions here:
18      https://developers.google.com/maps/documentation/android/start#get-key
19
20      Once you have your key (it starts with "AIza"), replace the "google_maps_key"
21      string in this file.
22      -->
23      <string name="google_maps_key" templateMergeStrategy="preserve"
24          translatable="false">AIzaSyCQZ4l9sgoHIi-yQHAdZfNnvHyzzYmGWhg</string>
25  </resources>
26
```

Build: | Build Output | Sync

- Build: build failed at 5/4/2022 5:34 PM — 1 m 37 s 3 ms
  - Run build E:\MapsAndroidDemo — 1 m 35 s 600 ms
    - Load build — 10 ms
    - Evaluate settings — 9 ms
      - Finalize build cache configuration
    - Configure build — 1 s 597 ms
    - Calculate task graph — 1 m 33 s 737 ms

Read timed out

TODO | Terminal | Build | 6: Logcat | Event Log

Gradle build finished in 1 m 37 s 80 ms (14 minutes ago)

26:1 CRLF UTF-8 4 spaces

Displaying the Map

To display Google Maps in your application, you first need the ACCESS_FINE_LOCATION permission in your manifest file.

This is created for you automatically when you selected to set up a Google Maps Activity. You can see the line if you open the AndroidManifest.xml.

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

Displaying the Zoom Control

The previous Demo showed how you can display Google Maps in your Android application.

You can pan the map to any desired location and it updates on-the-fly.

However, there is no way to use the emulator to zoom in or out from a particular location

Displaying the Zoom Control

- add the following bolded statement to activity_maps.xml.

```
<fragment
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:map="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/map"
tools:context="com.jfdimarzio.locationservices.MapsActivity"
android:name="com.google.android.gms.maps.SupportMapFragment"
map:uiZoomControls="true"
/>
```

Displaying the Zoom Control

- Besides displaying the zoom controls, you can also programmatically zoom in or out of the map using the animateCamera() method of the GoogleMap class.

MapsActivity

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

private GoogleMap mMap;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified
    // when the map is ready to be used.
    SupportMapFragment mapFragment = (SupportMapFragment)
    getSupportFragmentManager().findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}
```

MapsActivity

*Note: You don't have to call this method because it is called by default

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    // Add a marker in Sydney and move the camera
    LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new MarkerOptions().position(sydney).title(
    "Marker in Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

To put The Camera on Marker When it open...

MapsActivity

There is :
onKeyUp
onKeyPressed

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    switch (keyCode)
    {
        case KeyEvent.KEYCODE_3:
                mMap.animateCamera(CameraUpdateFactory.zoomIn());
        break;
        case KeyEvent.KEYCODE_1:
                mMap.animateCamera(CameraUpdateFactory.zoomOut());
        break;
    }
    return super.onKeyDown(keyCode, event);
}
}
```

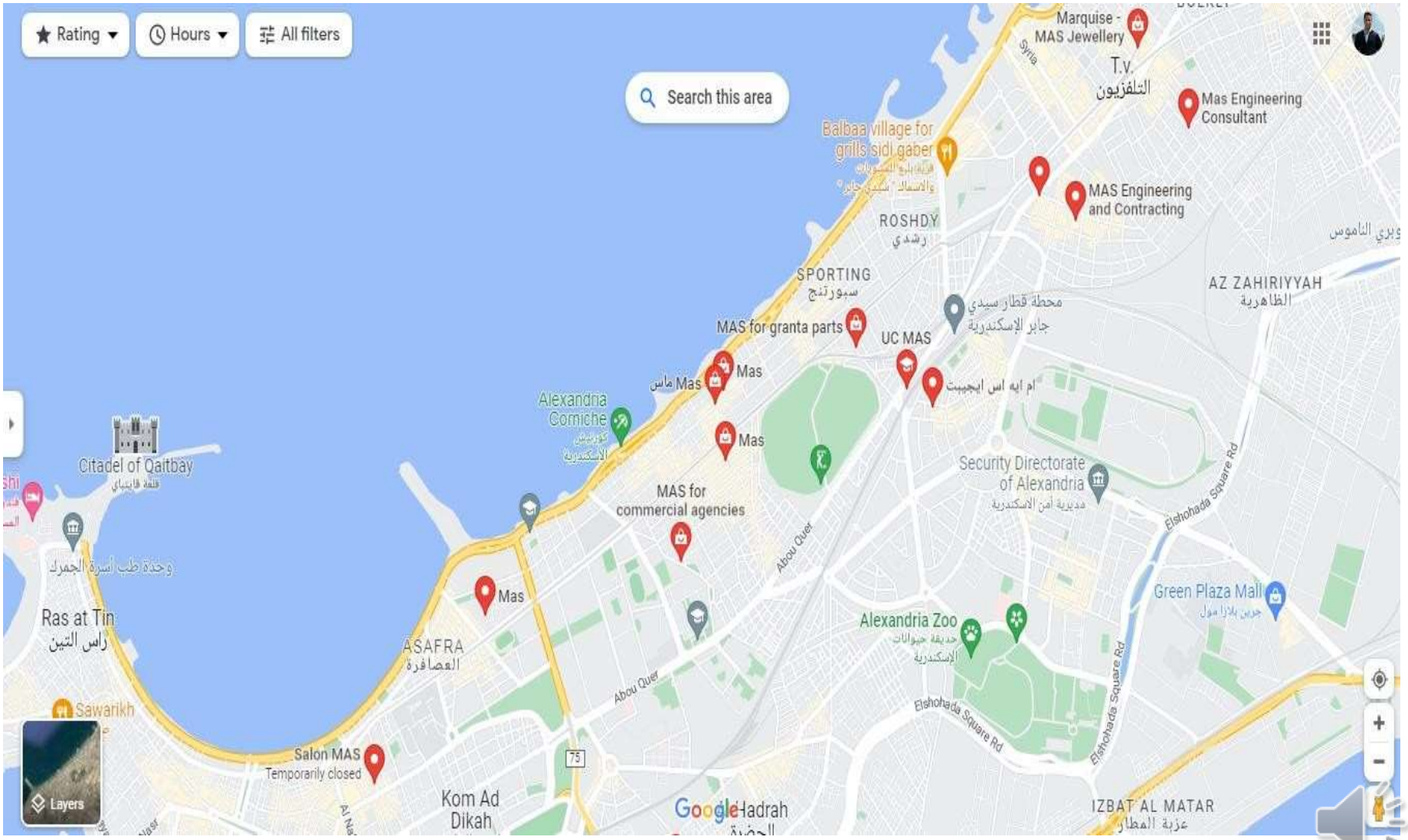
If The user click 3 from keyboard Zoom In

Changing Views

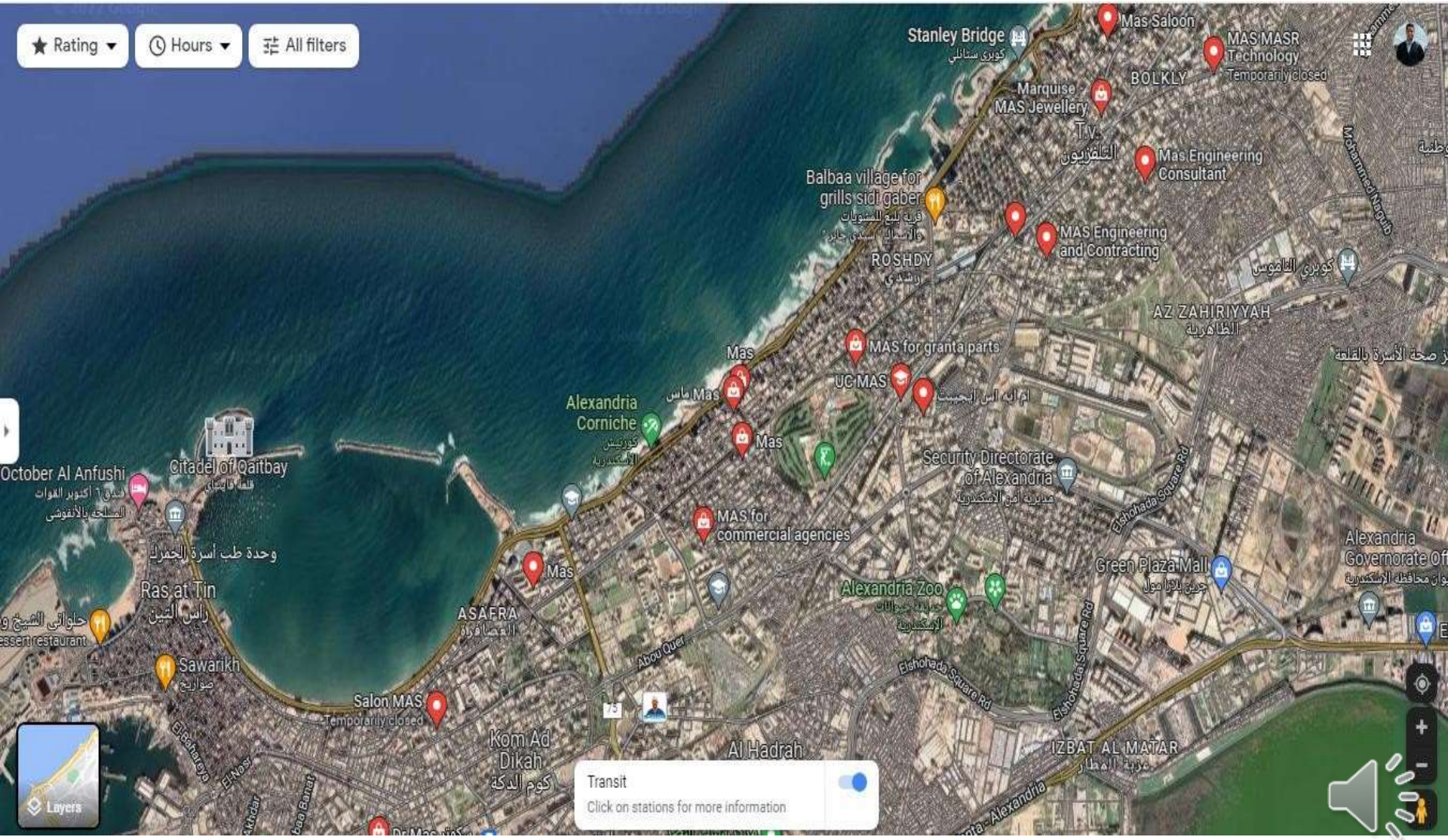
- By default, Google Maps is displayed in *map view*, which is basically drawings of streets and places of interest.

- You can also set Google Maps to display in *satellite view* using the **setMapType()** method of the GoogleMap class:

Changing Views



Changing Views



Changing Views

```
public void onMapReady(GoogleMap
googleMap) {
    mMap = googleMap;
    // Add a marker in Sydney and move the camera
    LatLng sydney = new LatLng(-34, 151);
    mMap.addMarker(new
    MarkerOptions().position(sydney).title(
    "Marker in Sydney"));
    mMap.moveCamera(CameraUpdateFactory.new
    LatLng(sydney));
    mMap.setMapType(GoogleMap.MAP_TYPE_SA
    TELLITE);
}
```

Navigating to a Specific Location

- By default, Google Maps displays the map of Australia when it is first loaded.

- However, you can set Google Maps to display a particular location.

- To do so, you can use the moveCamera() method of the GoogleMap class.

Navigating to a Specific Location

```
@Override
public void onMapReady(GoogleMap
googleMap) {
    mMap = googleMap;

    LatLng boston = new LatLng(42.3601, -
    71.0589);
    mMap.addMarker(new
    MarkerOptions().position(boston).title(
    "Boston, Mass"));
    mMap.moveCamera(CameraUpdateFactor
    y.newLatLng(boston));
}
```

Getting the Location That Was Touched

- you might want to know the latitude and longitude of a location corresponding to the position on the screen that was just touched

- Knowing this information is very useful because you can determine a location's address—a process known as reverse geocoding

- To get the latitude and longitude of a point on the Google Map that was touched, you must set a **onMapClickListener**

MapsActivity

```
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.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback
    private GoogleMap mMap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified
    // when the map is ready to be used.
    SupportMapFragment mapFragment =
    (SupportMapFragment)
    getSupportFragmentManager().findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}
```

MapsActivity

```
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;

LatLng boston = new LatLng(42.3601, -71.0589);
mMap.addMarker(new MarkerOptions().position(boston).title("Boston, Mass"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(boston));

mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
        @Override
        public void onMapClick(LatLng point) {
          Log.d("DEBUG","Map clicked [" + point.latitude + " / " + point.longitude + "]");
        }
    });
    }
}
```

Geocoding and Reverse Geocoding

- As mentioned in the preceding section, if you know the latitude and longitude of a location, you can find out its address using a process known as reverse geocoding.

- Google Maps in Android supports reverse geocoding via the **Geocoder class**.

MapsActivity

```
import android.location.Address;
import android.location.Geocoder;
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.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified
    // when the map is ready to be used.
    SupportMapFragment mapFragment =
    (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}
```

MapsActivity

```
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;
LatLng boston = new LatLng(42.3601, -71.0589);
mMap.addMarker(new MarkerOptions().position(boston).title("Boston, Mass"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(boston));
mMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {
        @Override
        public void onMapClick(LatLng point) {
          Geocoder geoCoder = new Geocoder(getBaseContext(), Locale.getDefault());
          try {
          List<Address> addresses = geoCoder.getFromLocation(point.latitude,point.longitude,1);
                    String add = "";
                    if (addresses.size() > 0){
                              for (int i=0; i<addresses.get(0).getMaxAddressLineIndex();i++)
                                        add += addresses.get(0).getAddressLine(i) + "\n";
                    }
        Toast.makeText(getBaseContext(), add, Toast.LENGTH_SHORT).show();
        }catch (IOException e) {
          e.printStackTrace();
        }} }); }
```

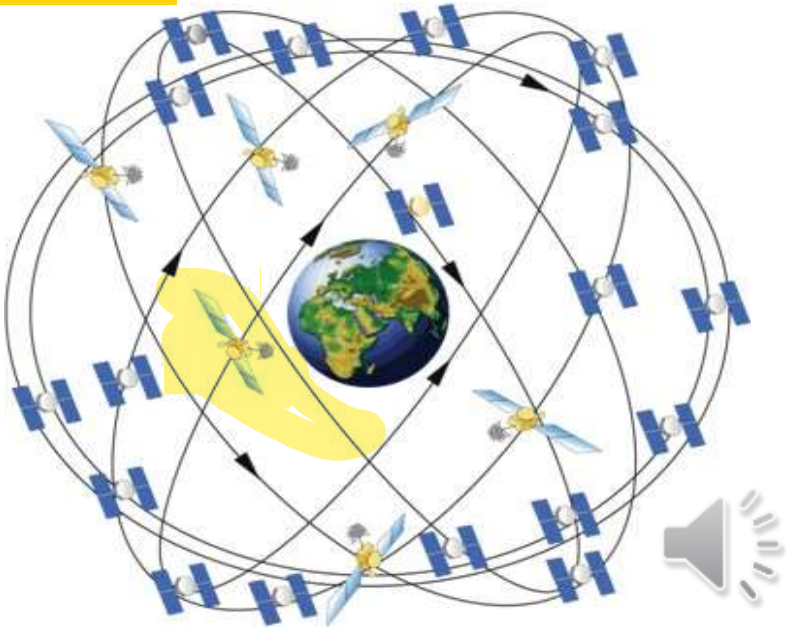
Geocoding and Reverse Geocoding

- If you know the address of a location but want to know its latitude and longitude, you can do so via geocoding. Again, you can use the Geocoder class for this purpose

```
Geocoder geoCoder = new Geocoder(getBaseContext(), Locale.getDefault());
try {
List<Address> addresses = geoCoder.getFromLocationName(
                                                "empire state building", 5);
if (addresses.size() > 0) {
    LatLng p = new LatLng(
            (int) (addresses.get(0).getLatitude()),
            (int) (addresses.get(0).getLongitude()));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(p));
}
} catch (IOException e) {
e.printStackTrace();
}
```

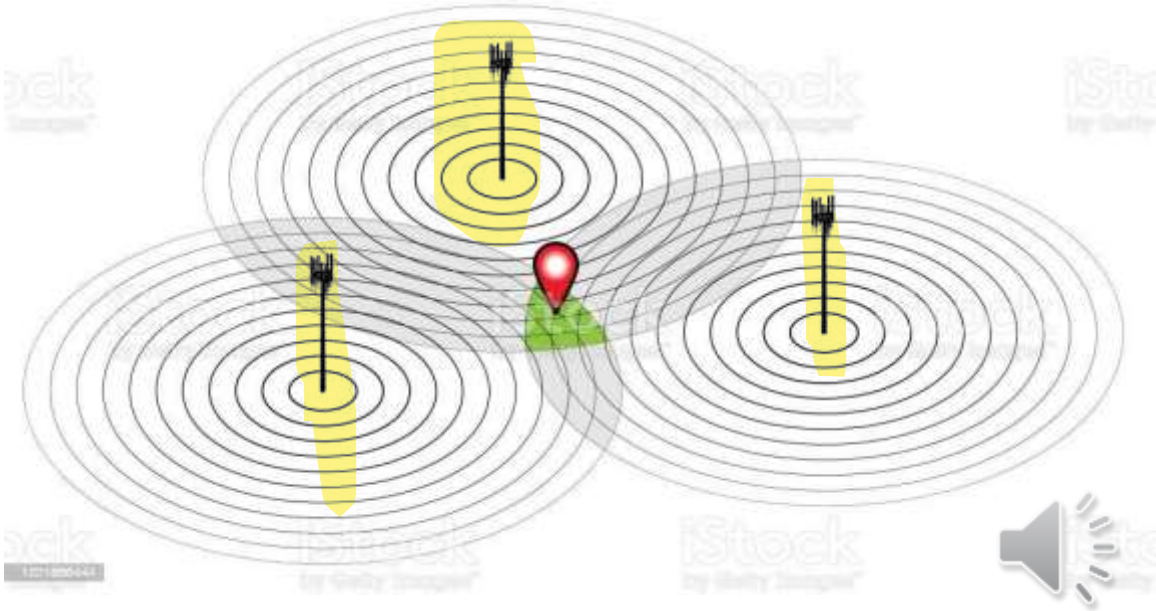
Getting Location Data

- Nowadays, mobile devices are commonly equipped with GPS receivers.
- you can use a GPS receiver to find your location easily
- GPS requires a clear sky to work and hence does not always work indoors or where satellites can't penetrate



Getting Location Data

- Another effective way to locate your position is through *cell tower triangulation*.
- However, it is not as precise as GPS because its accuracy depends on overlapping signal coverage,



Getting Location Data

- A third method of locating your position is to rely on Wi-Fi triangulation

- The device connects to a Wi-Fi network and checks the service provider against database to determine the location serviced by the provider.

- On the Android platform, the SDK provides the **LocationManager** class to help your device determine the user's physical location.

MapsActivity

```
//// Added list of import
public class MapsActivity extends FragmentActivity im
{
final private int REQUEST_COURSE_ACCESS = 123;
boolean permissionGranted = false;
private GoogleMap mMap;
LocationManager lm;
LocationListener locationListener;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    // Obtain the SupportMapFragment and get notified
    // when the map is ready to be used.
    SupportMapFragment mapFragment =
    (SupportMapFragment) getSupportFragmentManager()
    .findFragmentById(R.id.map);
    mapFragment.getMapAsync(this);
}
```

MapsActivity

```
@Override
public void onPause() {
    super.onPause();
    //---remove the location listener---
    if (ActivityCompat.checkSelfPermission(this,
        android.Manifest.permission.ACCESS_FINE_LOCATION)
    != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
    this, android.Manifest.permission.ACCESS_COARSE_LOCATION)
    != PackageManager.PERMISSION_GRANTED) {
      ActivityCompat.requestPermissions(this, new String[]{
        android.Manifest.permission.ACCESS_COARSE_LOCATION},
        REQUEST_COURSE_ACCESS);
        return;
    }else{
        permissionGranted = true;
    }
    if(permissionGranted) {
        lm.removeUpdates(locationListener);
}
}
```

GPS

Cell

wiFi

MapsActivity

```
@Override
public void onMapReady(GoogleMap googleMap) {
mMap = googleMap;
        lm = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
        locationListener = new MyLocationListener();
        if (ActivityCompat.checkSelfPermission(this,
        android.Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
        this, android.Manifest.permission.ACCESS_COARSE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,
                new String[]{android.Manifest.permission.ACCESS_COARSE_LOCATION},
                REQUEST_COURSE_ACCESS);
                return;
        }else{
                permissionGranted = true;
        }
    if(permissionGranted) {
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListener);
    }
}
```

be notified periodically

The **requestLocationUpdates()** method takes four arguments:

- provider—The name of the provider with which you register. In this case, you are using GPS to obtain your geographical location data.
- minTime—The minimum time interval for notifications, in milliseconds. 0 indicates that you want to be continually informed of location changes.
- minDistance—The minimum distance interval for notifications, in meters. 0 indicates that you want to be continually informed of location changes.
- listener—An object whose onLocationChanged() method will be called for each location update

MapsActivity

```
private class MyLocationListener implements LocationListener
{
    public void onLocationChanged(Location loc) {
        if (loc != null) {
        Toast.makeText(getBaseContext(), "Location changed : Lat: " + loc.getLatitude() +
        " Lng: " + loc.getLongitude(), Toast.LENGTH_SHORT).show();
        LatLng p = new LatLng( (int) (loc.getLatitude()), (int) (loc.getLongitude()));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(p));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(7));
    }
}
public void onProviderDisabled(String provider) {
        Toast.makeText(getBaseContext(), provider + " disabled", Toast.LENGTH_SHORT).show();
}
public void onProviderEnabled(String provider) {
}
public void onStatusChanged(String provider, int status, Bundle extras) {
} } }
```

Add the following bolded line to the AndroidManifest.xml file.

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

Getting Location Data

- You can combine both the GPS location provider with the network location provider within your application:

```
//---request for location updates---
lm.requestLocationUpdates( LocationManager.GPS_PROVIDER,0, 0, locationListener);

//---request for location updates---
lm.requestLocationUpdates( LocationManager.NETWORK_PROVIDER,0,0,locationListener);
```

End of Lecture