CS 193A

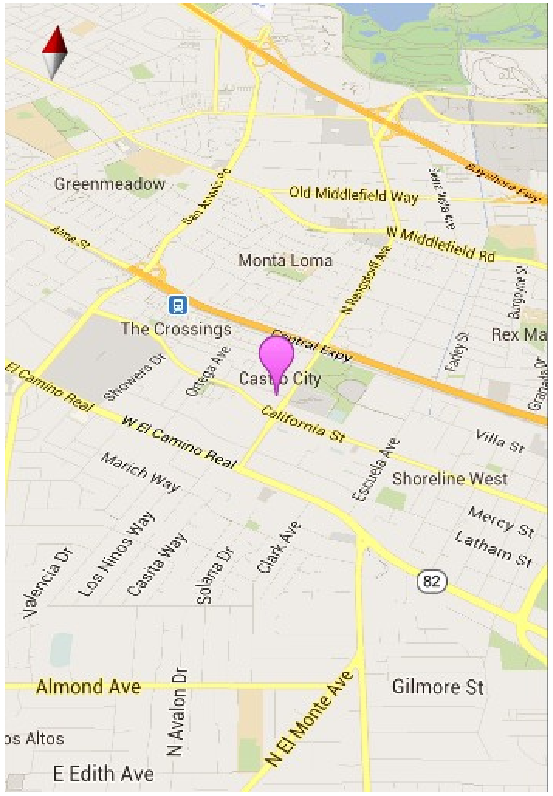
Maps, Location, and GPS

Maps and location

- Many apps use a **map view** to show a display of a region.
  - Typically implemented as a **fragment** called a `MapFragment`.

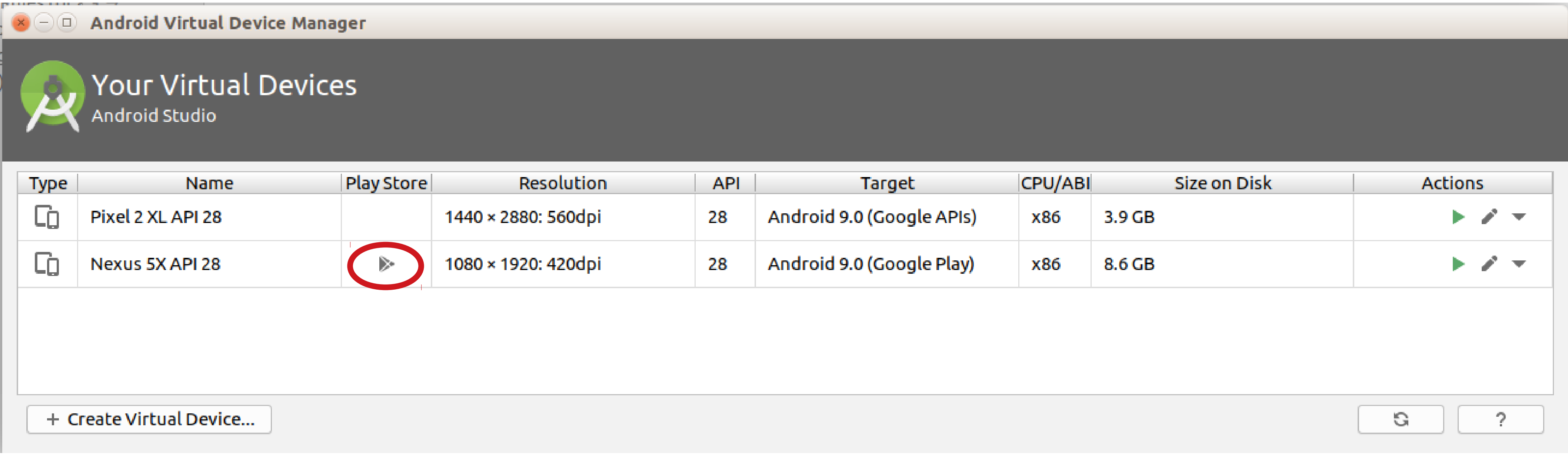
- Some apps also display information about the user's **location**.
  - Can ask phone for user's coordinates.
  - Can display user on map as they move around.

- Maps are super cool!
  - Sadly, there is some complexity to discuss.
  - We will need **API keys** and various configuration of our project.
  - The code to use maps has a few quirks.



Emulator Google Play services

- need to use an emulator device w/ **Google Play** services
  - run AVD Manager to check your device



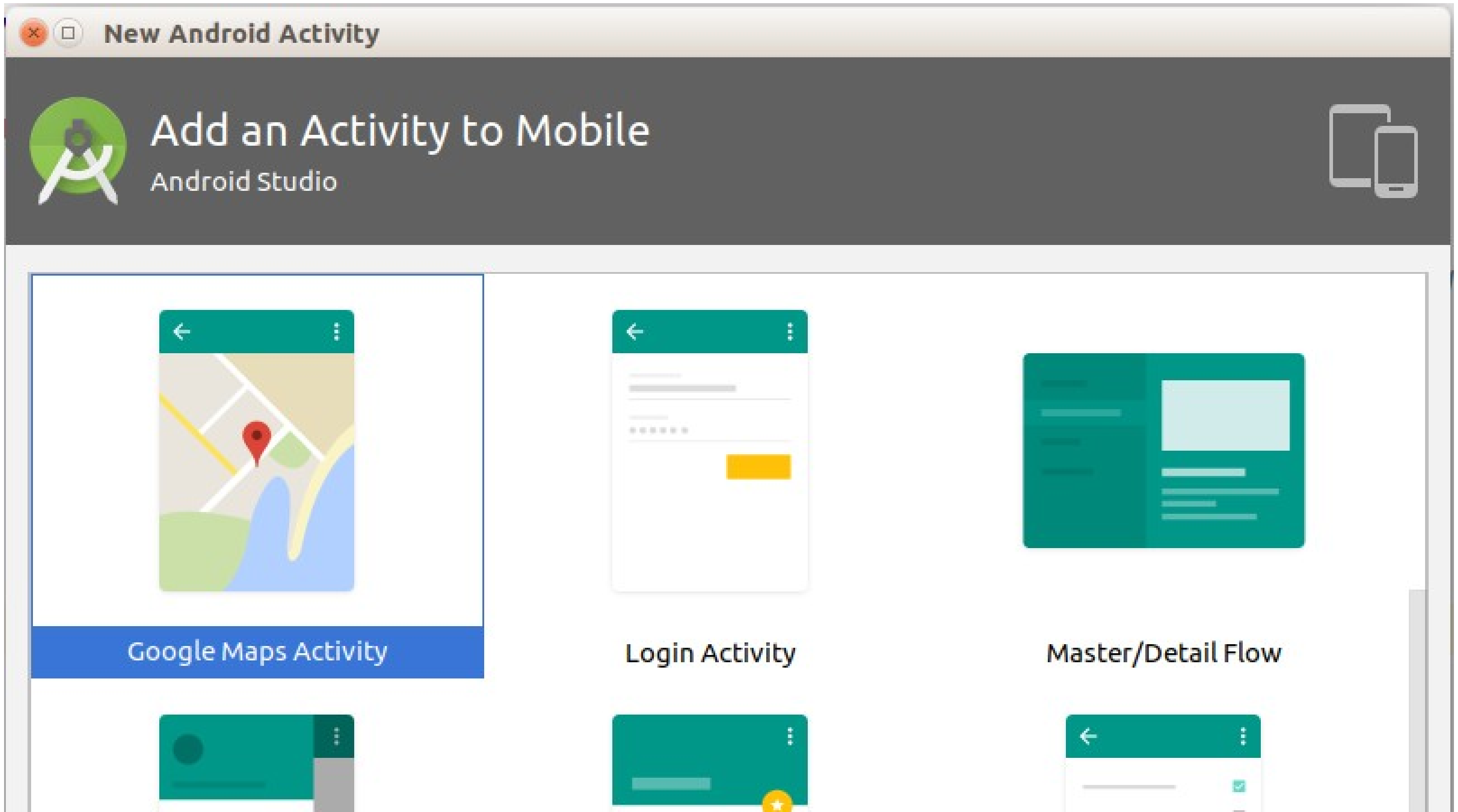
Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 2 XL API 28		1440 × 2880: 560dpi	28	Android 9.0 (Google APIs)	x86	3.9 GB	▶ ✎ ▾
	Nexus 5X API 28	▶	1080 × 1920: 420dpi	28	Android 9.0 (Google Play)	x86	8.6 GB	▶ ✎ ▾

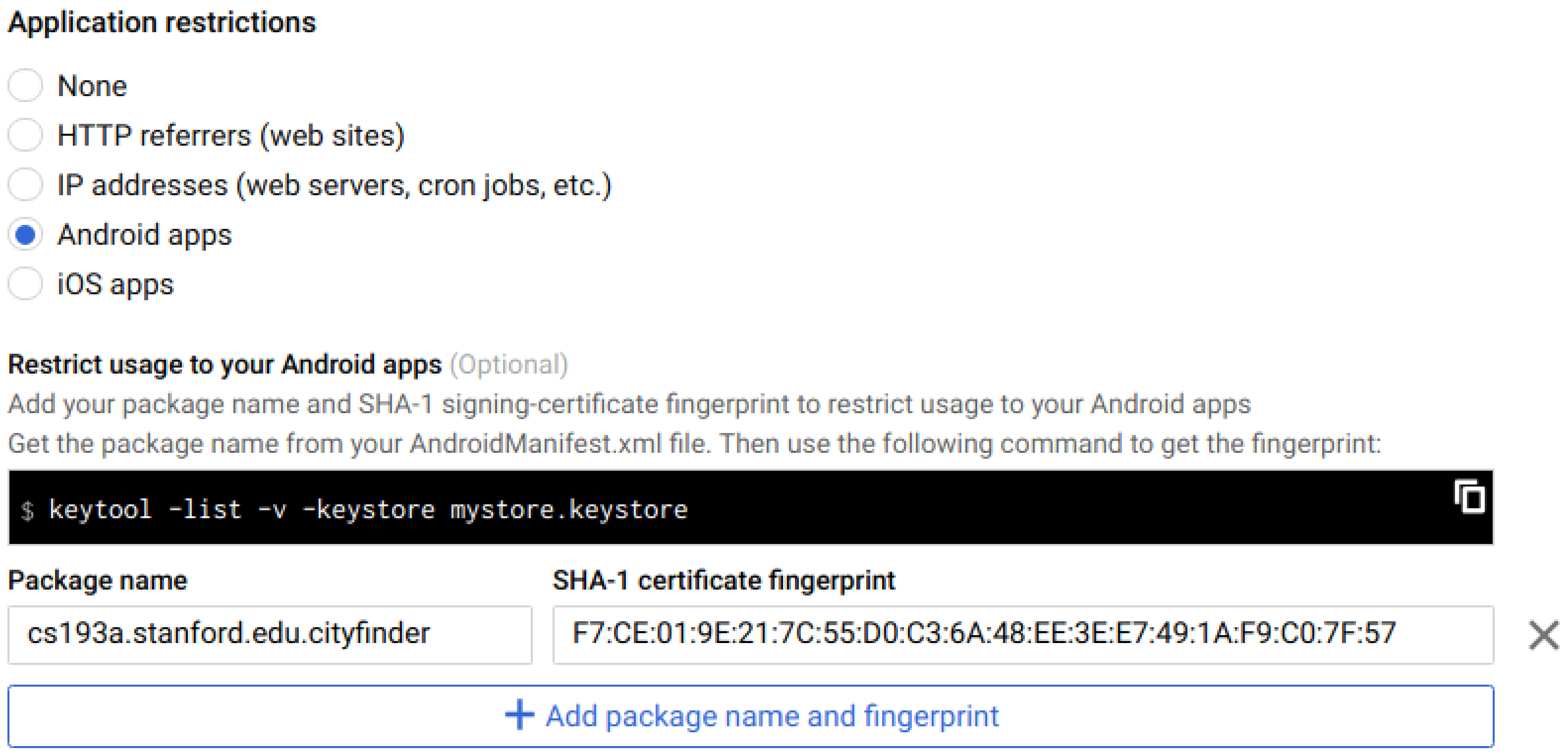
Adding a map to your app

- In Android Studio, right-click your project's package
- click Add → Activity → Gallery ... → Google Maps Activity
  - this will make lots of changes to your project config;
    see slides at end if you need to make/alter these changes manually



Maps API keys

- Go to the Google APIs developer console:
  - https://console.developers.google.com/
  - under APIs & Services, click Credentials → Create credentials → **API key**
  - set Application **Restrictions** to: Android apps
  - paste this key back into your app's code in appropriate **XML files**



Application restrictions

- None
- HTTP referrers (web sites)
- IP addresses (web servers, cron jobs, etc.)
- [✓ Android apps]
- iOS apps

Restrict usage to your Android apps (Optional)

Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps

Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

```
$ keytool -list -v -keystore mystore.keystore
```

Package name	SHA-1 certificate fingerprint
cs193a.stanford.edu.cityfinder	F7:CE:01:9E:21:7C:55:D0:C3:6A:48:EE:3E:E7:49:1A:F9:C0:7F:57

MapFragment (link)

- Google Maps API provides a fragment class named **`MapFragment`** for displaying a map within an activity.

```
<android.support.constraint.ConstraintLayout ...
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:map="http://schemas.android.com/apk/res-auto"
  tools:ignore="MissingPrefix">

  <fragment ...
      android:name="com.google.android.gms.maps.MapFragment"
      android:id="@+id/ID" />
```

  - *(There is also a MapView class that we won't cover)*

Waiting for map to be ready

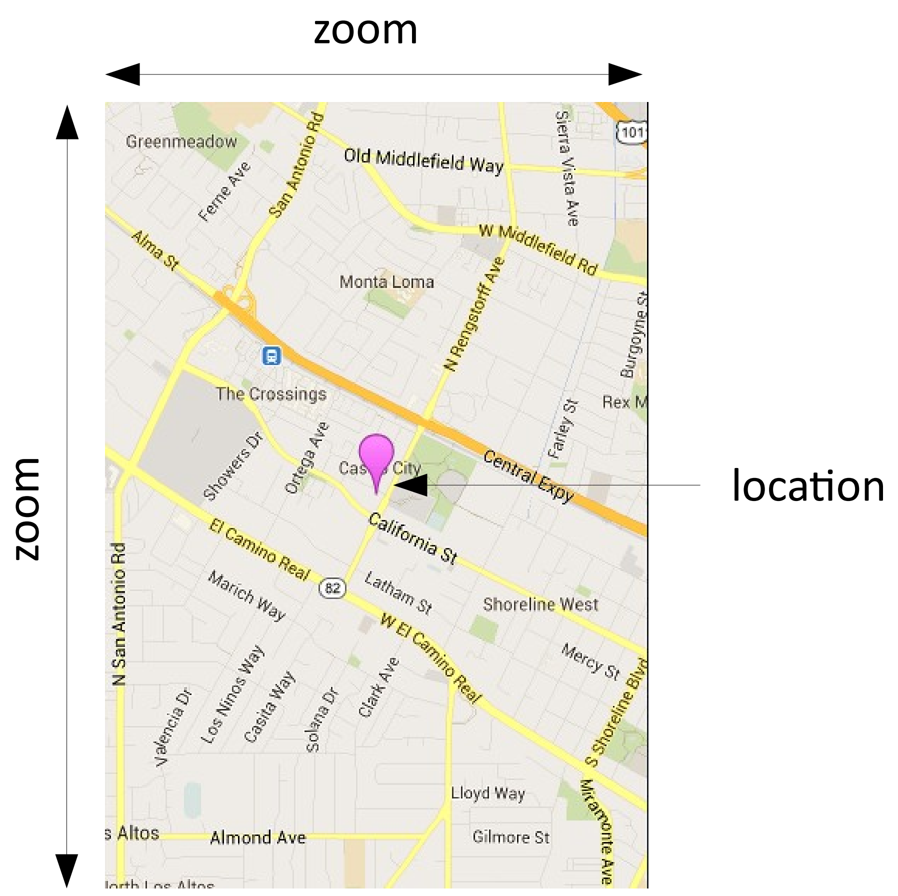
```
class Name : AppCompatActivity(), OnMapReadyCallback {
    private lateinit var map : GoogleMap

    override fun onCreate(bundle: Bundle?) {
        ...
        val mf = fragmentManager.findFragmentById(R.id.ID)
                 as MapFragment
        mf.getMapAsync { googleMap ->
            map = googleMap

            // code to run when the map is ready ...
        }
    }
}
```

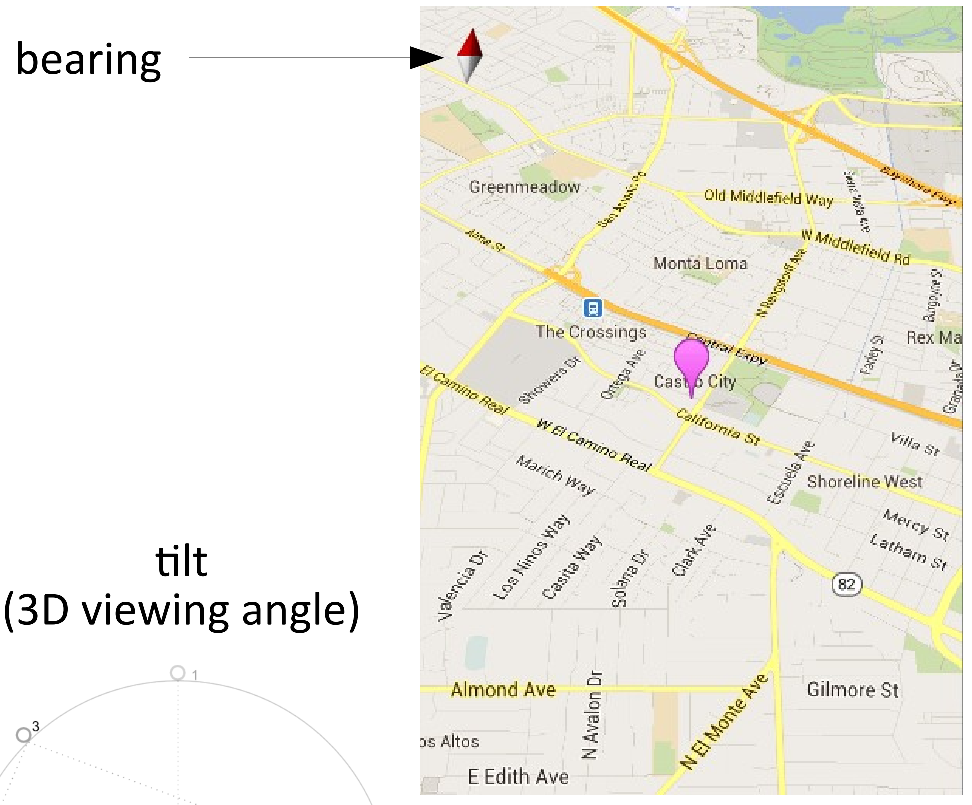
GoogleMap methods (link)

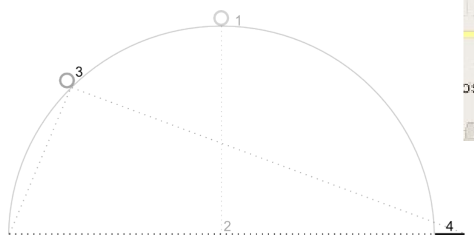
- placing items on the map:
  - addCircle, addGroundOverlay, **addMarker**, addPolygon, **addPolyline**, addTileOverlay
  - **clear** - Removes all markers, polylines/polygons, overlays
- manipulating the camera:
  - cameraPosition, **moveCamera**, **animateCamera**, stopAnimation
- map settings and appearance:
  - buildingsEnabled, indoorEnabled, mapType, padding, trafficEnabled
- snapshot - take a screen shot of the map as a bitmap
- event listeners:
  - setOnCameraChangeListener, **setOnMapClickListener**, setOnMapLoadedCallback, setOnMapLongClickListener, **setOnMarkerClickListener**, setOnMarkerDragListener, setOnMyLocationChangeListener

The map's camera

- The current viewing window of a map's camera is defined by:
  - **target** location (latitude/longitude), **zoom** (2.0 - 21.0),
  - **bearing** (orientation/rotation), and **tilt** (degrees)



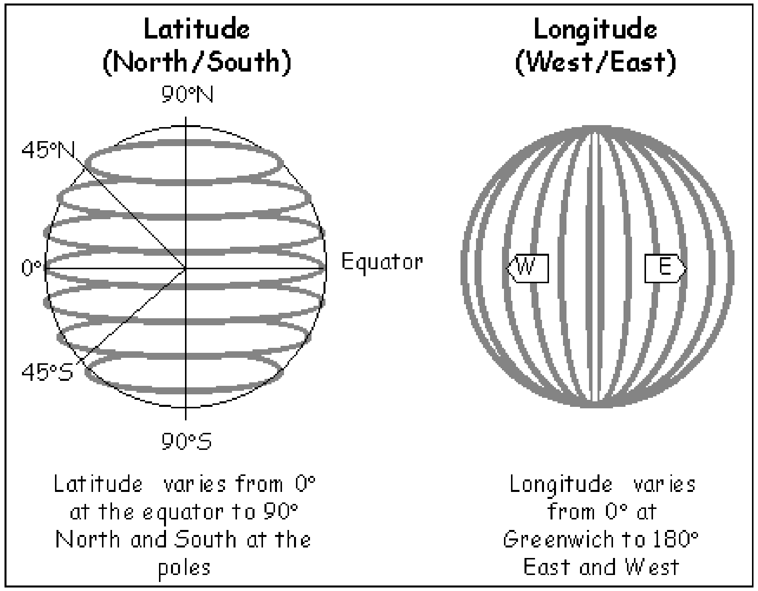


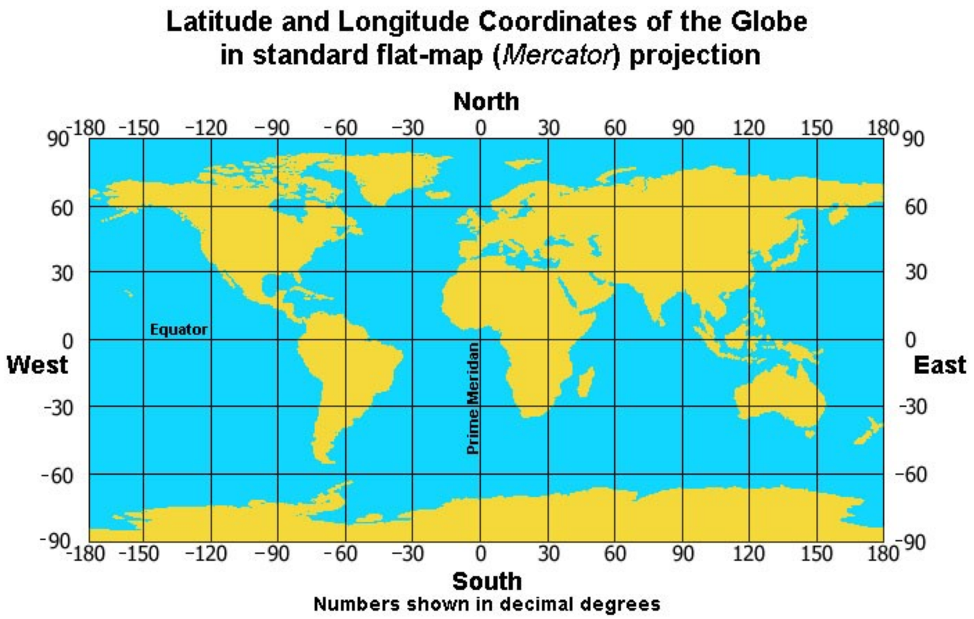


Latitude and longitude

- **latitude**: N/S angle relative to the equator
  - North pole = +90; South pole = -90
- **longitude**: E/W angle relative to prime meridian
  - West = 0 → -180;   East = 0 → 180
  - *find lat/lng of a place on Google Maps in URL address bar*

    see also: http://itouchmap.com/latlong.html

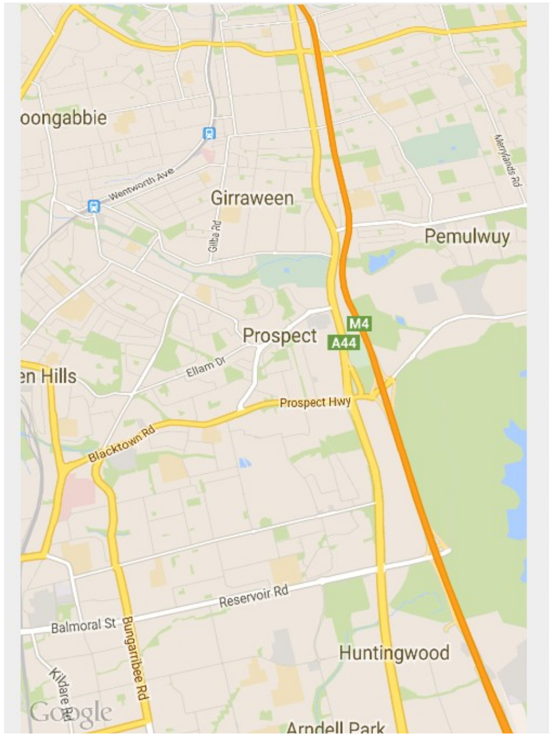

Latitude
(North/South)
90°N
45°N
0°
Equator
45°S
90°S
Latitude varies from 0° at the equator to 90° North and South at the poles
Longitude
(West/East)
W
E
Longitude varies from 0° at Greenwich to 180° East and West


Latitude and Longitude Coordinates of the Globe
in standard flat-map (Mercator) projection
North
South
West
East
Equator
Prime Meridian
-180 -150 -120 -90 -60 -30 0 30 60 90 120 150 180
90 60 30 0 -30 -60 -90
Numbers shown in decimal degrees

Set camera in XML

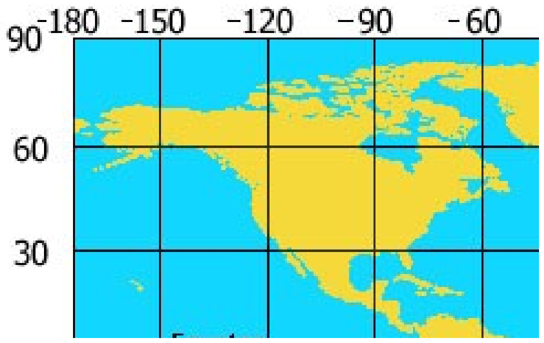
- Set initial map settings and camera position in the layout XML:
  - see here (link) for full list of attributes available

```
<fragment ...
    android:name="com.google.android.gms.maps.MapFragment"
    android:id="@+id/ID"
    map:cameraBearing="112.5"
    map:cameraTargetLat="-33.796923"
    map:cameraTargetLng="150.922433"
    map:cameraTilt="30"
    map:cameraZoom="13"
    map:mapType="normal"
    map:uiCompass="false"
    map:uiRotateGestures="true"
    map:uiScrollGestures="false"
    map:uiTiltGestures="true"
    map:uiZoomControls="false"
    map:uiZoomGestures="true" />
```



Set camera in Kotlin code (link)

- `CameraUpdateFactory` methods:
  - `newLatLng(LatLng(`***`lat`***`, `***`lng`***`))`
  - `newLatLngBounds(LatLngBounds(`***`SW`***`, `***`NE`***`), `***`padding`***`)`
  - `newLatLngZoom(LatLng(`***`lat`***`, `***`lng`***`), `***`zoom`***`)`
  - `newCameraPosition(`***`CameraPosition`***`)`
  - `others:`

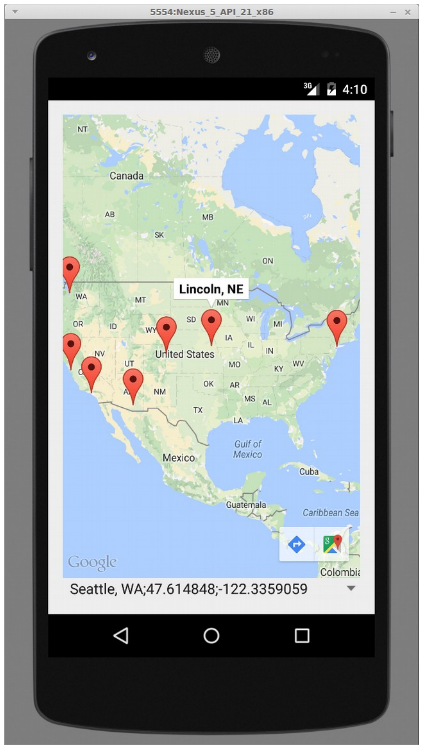


```
// example; show roughly the entire USA
val bounds = LatLngBounds(
    LatLng(20, -130.0),     // SW
    LatLng(55,  -70.0))    // NE

map.moveCamera(CameraUpdateFactory.newLatLngBounds(bounds, 50))

// try also: map.animateCamera
```

Placing markers

- A `GoogleMap` object has an **`addMarker`** method that can let you put "push pin" markers at locations on the map.
  - The marker's methods return the marker, so you can chain them.
  - options: alpha, draggable, icon, position, rotation, title, visible, ...

```
map.addMarker(MarkerOptions()
    .position(LatLng(40.801, -96.691))
    .title("Lincoln, NE")
)

// to modify/remove the marker later,
// save it as a variable
val mark = map.addMarker(MarkerOptions()
    ...)
mark.remove()
```

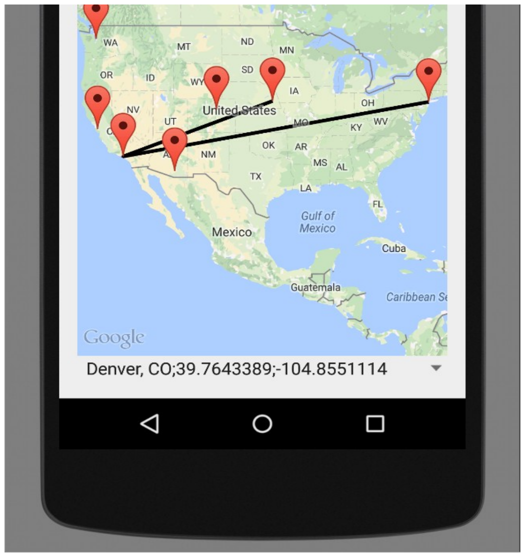


Seattle, WA;47.614848;-122.3359059

Lines and paths

- A `GoogleMap` object has an **`addPolyline`** method that can let you put lines between locations on the map.
  - options: color, visible, width, zIndex, ...

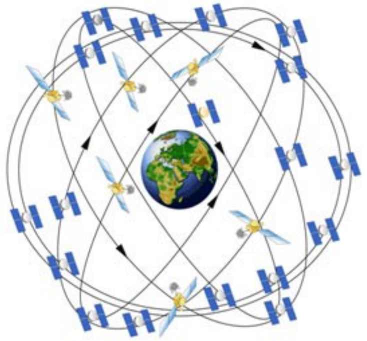
```
map.addPolyline(PolylineOptions()
    .add(LatLng(40.801, -96.691))    // Lincoln, NE
    .add(LatLng(34.020, -118.412))   // Los Angeles, CA
    .add(LatLng(40.703, -73.980))    // New York, NY
);

// to modify/remove the line later,
// save it as a variable
val polly = map.addPolyline(...)
polly.remove()
```



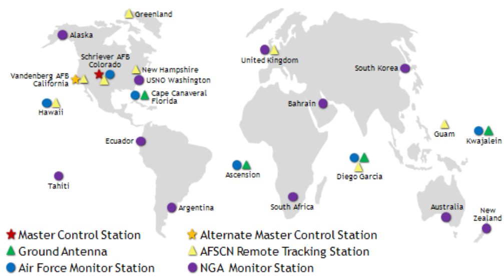
Denver, CO;39.7643389;-104.8551114

Global Positioning System (GPS)

- US System that provides position, navigation, and timing
- Space Segment
  - 27 core satellites developed by Lockheed Martin (FOCS)
  - 6 orbit planes with 4 satellites each; circle earth 2x/day
  - generally 4 satellites in line of sight at any spot on earth



- Control Segment
  - ground facilities operated by US Air Force
  - monitor, analyze, send commands/data to satellites



- User Segment
  - accurate within ~5 - 10 meters
  - military broadcasts on two freqs (more accurate); civilian only one

Accessing phone's location (link)

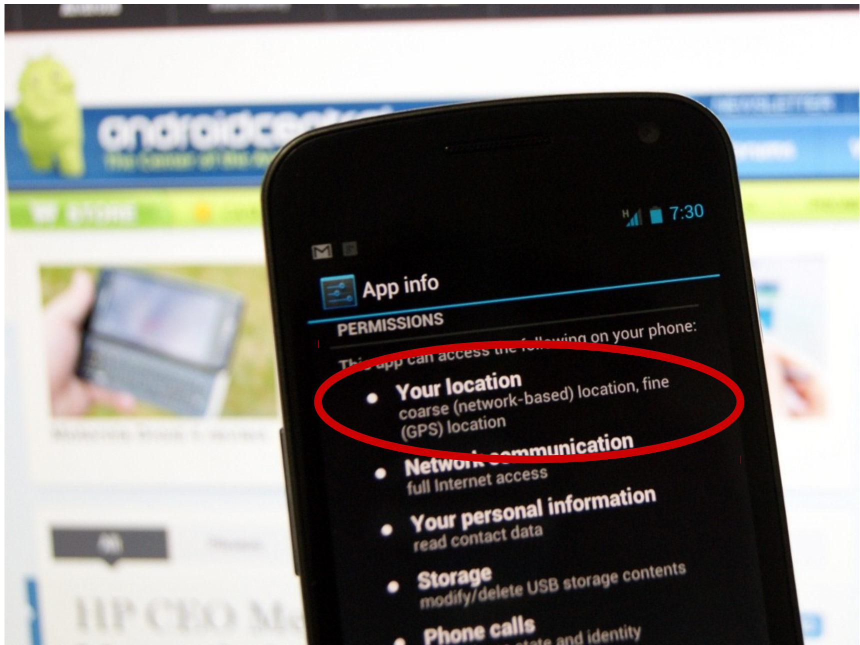
- Android **LocationManager** gives you the phone's position:
  - GPS provider provides highest accuracy
  - Network provider is a fallback in case GPS is disabled / not present

```
val locMgr = getSystemService(LOCATION_SERVICE) as LocationManager
val loc = locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER)
if (loc == null) {   // fall back to network if GPS is not available
    loc = locMgr.getLastKnownLocation(LocationManager.NETWORK_PROVIDER)
}
if (loc == null) {   // fall back to 'passive' location
    loc = locMgr.getLastKnownLocation(LocationManager.PASSIVE_PROVIDER)
}
if (loc != null) {
    val lat = loc.latitude
    val lng = loc.longitude   // Double
    ...
    // other properties: altitude, speed, bearing, ...
}
```

AndroidManifest.xml changes

- Because of privacy issues, to access phone's current location, must request permission in **AndroidManifest.xml**:

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

   <application ...>
       ...
   </application>
</manifest>
```



App info

PERMISSIONS

This app can access the following on your phone:

- **Your location** coarse (network-based) location, fine (GPS) location
- **Network communication** full Internet access
- **Your personal information** read contact data
- **Storage** modify/delete USB storage contents
- **Phone calls**

Permissions in Kotlin (link)

- In Android 6 they changed the permission model.
- Instead of just asking for perms in Manifest XML, you must also ask for them in the Kotlin code:

```
val perm = ActivityCompat.checkSelfPermission(this,
           Manifest.permission.PERMISSION_NAME)
if (perm == PackageManager.PERMISSION_GRANTED) {
    // you have permission! gogogo
    // ...
} else {
    // We don't have permission, so ask the user
    // (will call onRequestPermissionsResult when done)
    ActivityCompat.requestPermissions(this,
            arrayOf(Manifest.permission.PERMISSION_NAME),
            REQUEST_CODE)
}
```

Faking emu location w/ terminal

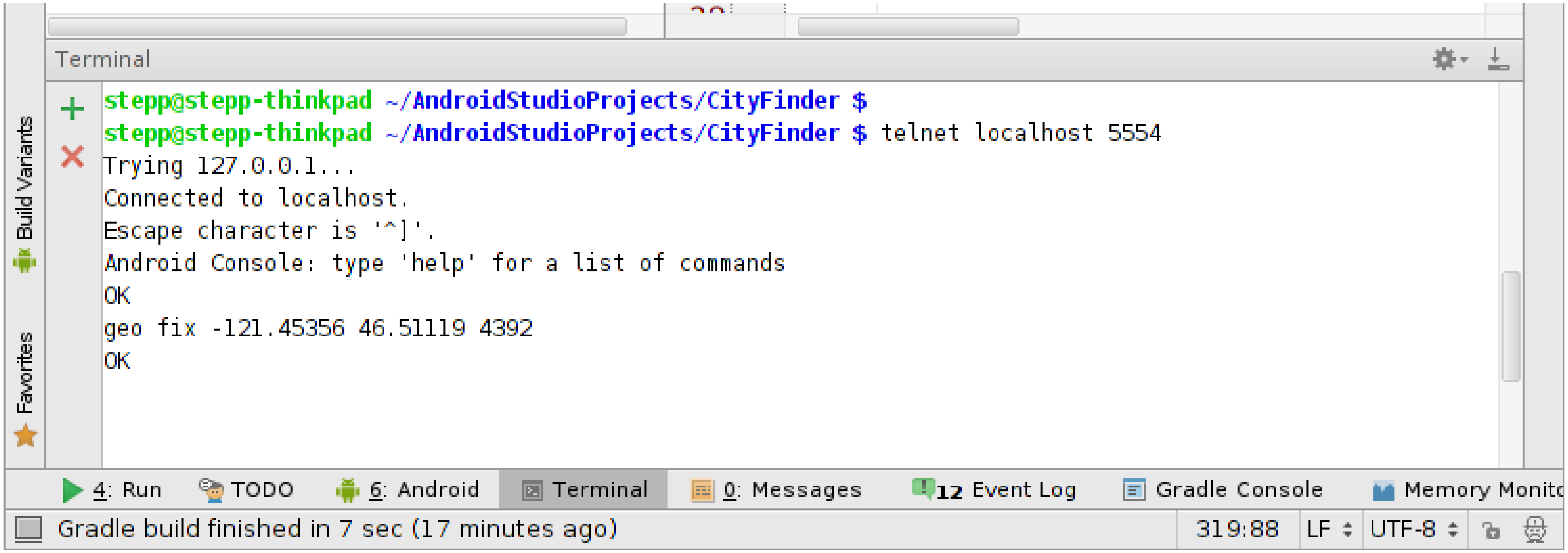
- Open a **Terminal**, and type:

  `telnet localhost 5554`

- once connected, type: *(altitude is optional)*

  auth ***your_auth_token***

  geo fix ***latitude longitude altitude***



```
stepp@stepp-thinkpad ~/AndroidStudioProjects/CityFinder $
stepp@stepp-thinkpad ~/AndroidStudioProjects/CityFinder $ telnet localhost 5554
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Android Console: type 'help' for a list of commands
OK
geo fix -121.45356 46.51119 4392
OK
```

Location update events

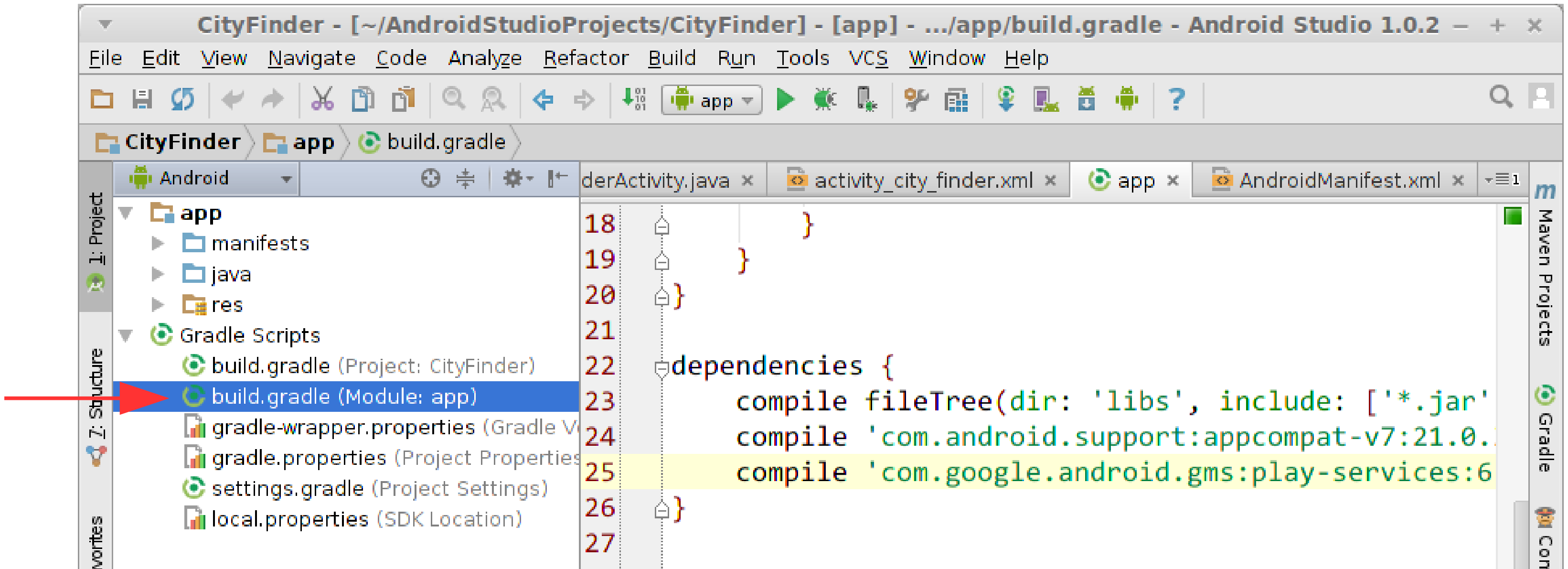
- Track user's movement by listening for location update events.

```
val locMgr = getSystemService(LOCATION_SERVICE) as LocationManager
locMgr.requestLocationUpdates(
    LocationManager.GPS_PROVIDER,
    minTime, minDistance,         // 0L/0F for as frequently as possible
    object : LocationListener() {
        override fun onLocationChanged(location: Location?) {
            // code to run when user's location changes

        }
        override fun onStatusChanged(prov: String?, status: Int,
                                     extras: Bundle?) {}
        public void onProviderEnabled(provider: String?) {}
        public void onProviderDisabled(provider: String?) {}
    }
);
```

Adding Play Services manually

- if you don't add a map activity through the A.Studio UI, you must add Google Play to project in app's **build.gradle** file:

```
dependencies {
  ...
  implementation 'com.google.android.gms:play-services-base:16.1.0'
  implementation 'com.google.android.gms:play-services-location:16.0.0'
  implementation 'com.google.android.gms:play-services-maps:16.1.0'
}
```


CityFinder - [~/AndroidStudioProjects/CityFinder] - [app] - .../app/build.gradle - Android Studio 1.0.2
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CityFinder app build.gradle
Android
app
manifests
java
res
Gradle Scripts
build.gradle (Project: CityFinder)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle V
gradle.properties (Project Properties
settings.gradle (Project Settings)
local.properties (SDK Location)
derActivity.java
activity_city_finder.xml
app
AndroidManifest.xml
1: Project
7: Structure
Maven Projects
Gradle

```
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'
    compile 'com.android.support:appcompat-v7:21.0.
    compile 'com.google.android.gms:play-services:6
}
```

Get an API key, part 1

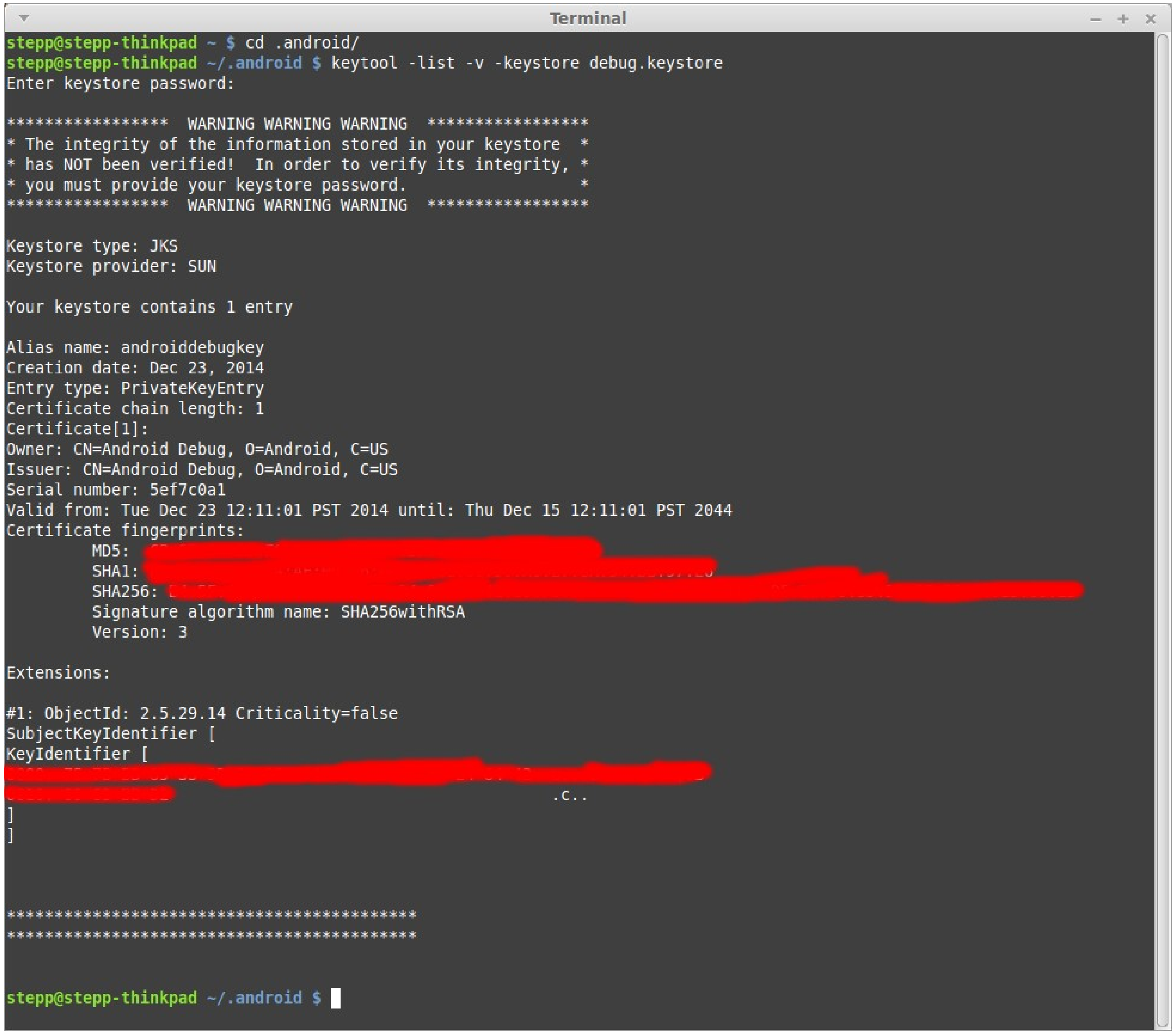
- Google won't allow you to fetch map data without an **API key**.
- To get a key, open a Terminal and find the file **debug.keystore**:
  - Windows: C:\Users\USERNAME\\.android
  - Linux: /home/*USERNAME*/.android/
  - Mac: /Users/*USERNAME*/.android/ (?)
- In the terminal, **cd** to that directory, then type:

```
keytool -list -v -keystore debug.keystore
```

  (it asks for a password, so just press Enter)

- Find the line with your "Certificate fingerprint" for "SHA-1".
  It should contain a long string in this format. Copy it down.
  - BD:2B:3F:4B:.........

Get an API key, part 1 (screenshot)

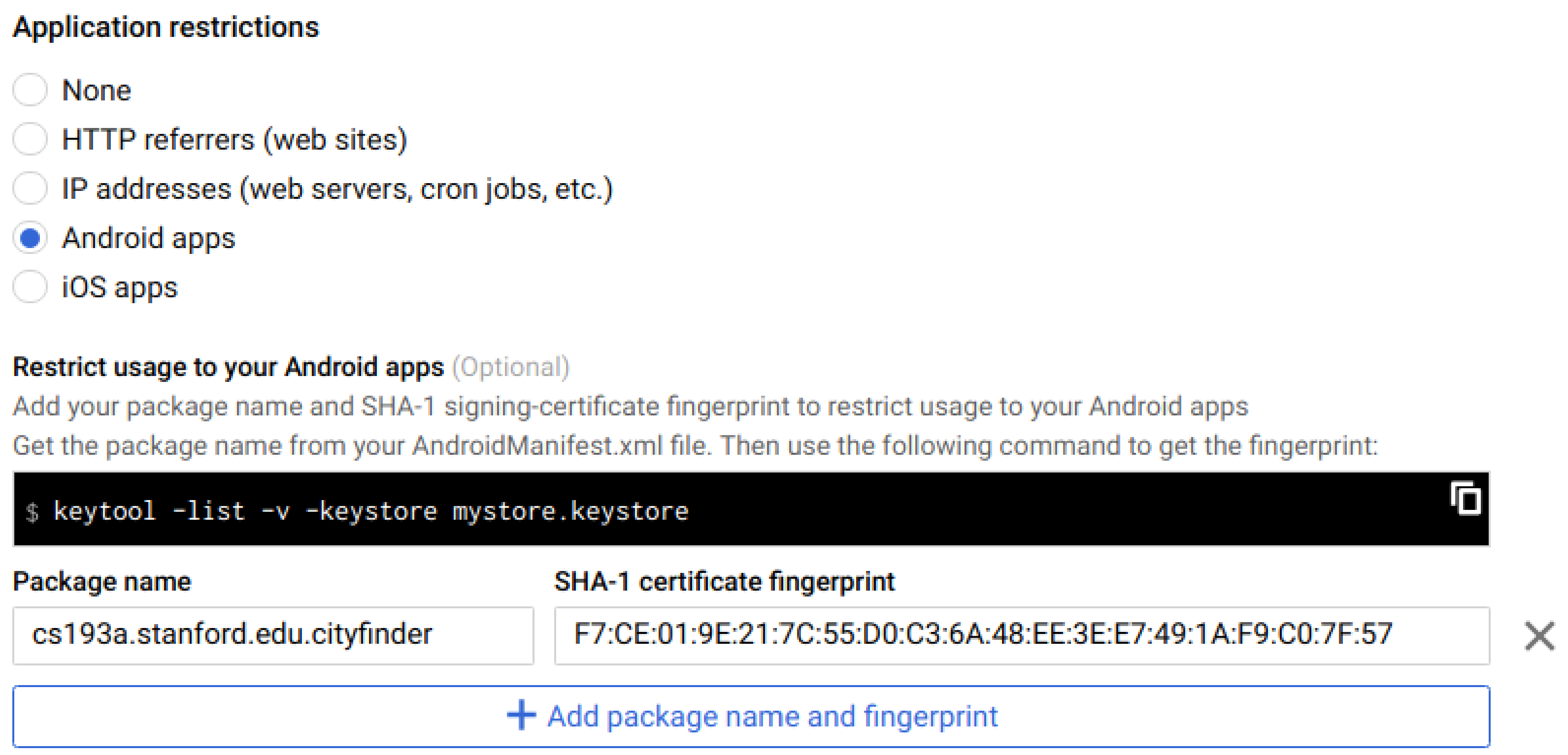


Terminal

```
stepp@stepp-thinkpad ~ $ cd .android/
stepp@stepp-thinkpad ~/.android $ keytool -list -v -keystore debug.keystore
Enter keystore password:

*****************  WARNING WARNING WARNING  *****************
* The integrity of the information stored in your keystore  *
* has NOT been verified!  In order to verify its integrity, *
* you must provide your keystore password.                  *
*****************  WARNING WARNING WARNING  *****************

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: androiddebugkey
Creation date: Dec 23, 2014
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 5ef7c0a1
Valid from: Tue Dec 23 12:11:01 PST 2014 until: Thu Dec 15 12:11:01 PST 2044
Certificate fingerprints:
         MD5:  [redacted]
         SHA1: [redacted]
         SHA256: [redacted]
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
[redacted]
[redacted]                                              .c..
]
]



*******************************************
*******************************************


stepp@stepp-thinkpad ~/.android $ 
```

Get an API key, part 2

- Go to the Google APIs developer console:
  - https://console.developers.google.com/
  - under APIs & Services, click Credentials → Create credentials → API key
  - set Application Restrictions to: Android apps
  - enter your project's package name and SHA-1 key from terminal



Application restrictions

None
HTTP referrers (web sites)
IP addresses (web servers, cron jobs, etc.)
[✓ Android apps]
iOS apps

Restrict usage to your Android apps (Optional)
Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps
Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:

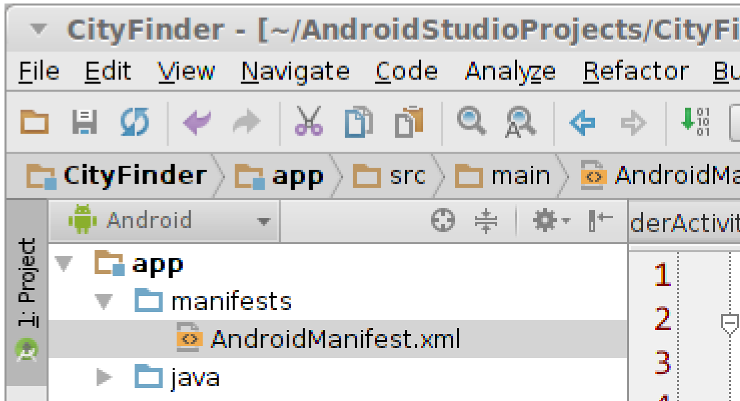
```
$ keytool -list -v -keystore mystore.keystore
```

Package name: cs193a.stanford.edu.cityfinder
SHA-1 certificate fingerprint: F7:CE:01:9E:21:7C:55:D0:C3:6A:48:EE:3E:E7:49:1A:F9:C0:7F:57

AndroidManifest.xml changes

- To use maps in your app, must make some manifest changes:

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

   <application ...>
       <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="your API key" />

       <activity ...> ... </activity>
   </application>
</manifest>
```


CityFinder - [~/AndroidStudioProjects/CityFi
File Edit View Navigate Code Analyze Refactor Bu
CityFinder > app > src > main > AndroidMa
Android
1: Project
app
manifests
AndroidManifest.xml
java
derActivit