Location

Overview

- Location Services
- Geocoding

Location Services

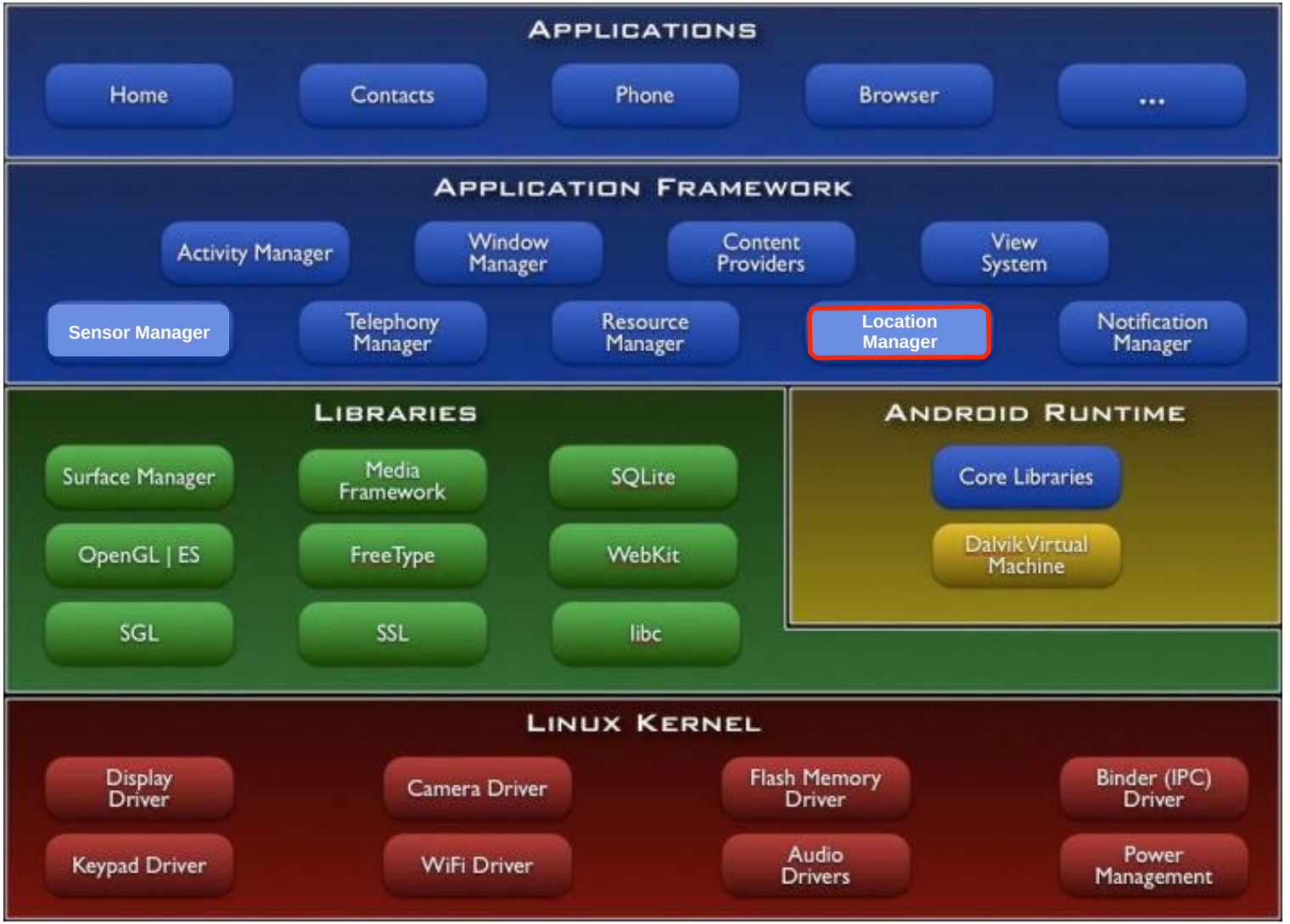
Location Service

- **Two main LBS elements**
  - **Location Manager** Provides hooks to the location-based services
  - **Location Providers** Each of these represents a different location-finding technology used to determine the device's current location

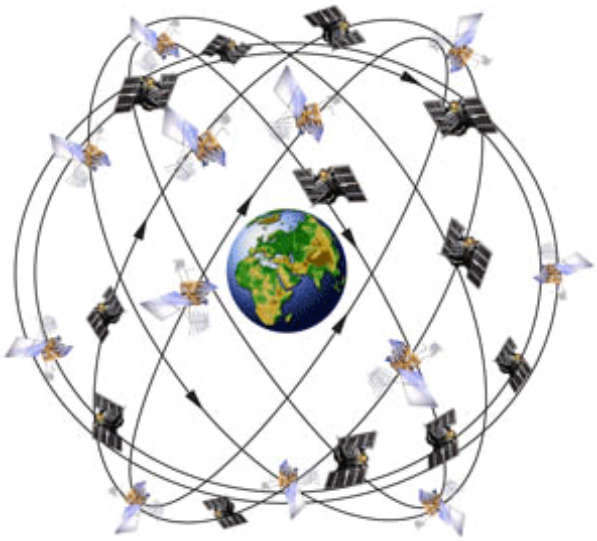
- **Location Manager**
  - Obtain current location
  - Track movement
  - Set proximity alerts for areas
  - Find available Location Providers

- **Location Providers**
  - Various location-finding technologies (GPS, Cellular network)

Android Software Stack



Global Positioning System (GPS)



The Global Positioning System (GPS) consists of 27 Earth-orbiting satellites (24 in operation and three extras in case one fails).

Developed by the USA as a military navigation system, but soon it opened to other civilian uses.

Each of these 3000- to 4000-pound solar powered satellites circles the globe at about 12000 miles, making two complete rotations every day.

The orbits are arranged so that at any time, anywhere on earth, there are at least four satellites visible in the sky.

A GPS receiver's job is to locate three or more of these satellites, figure out the distance to each, and use this information to deduce its own location. This operation is based on a mathematical principle called **trilateration.**

Global Positioning System (GPS)

Trilateration

--- Miami1795 km

--- Caracas 1874 km

--- Bogota 1251 km

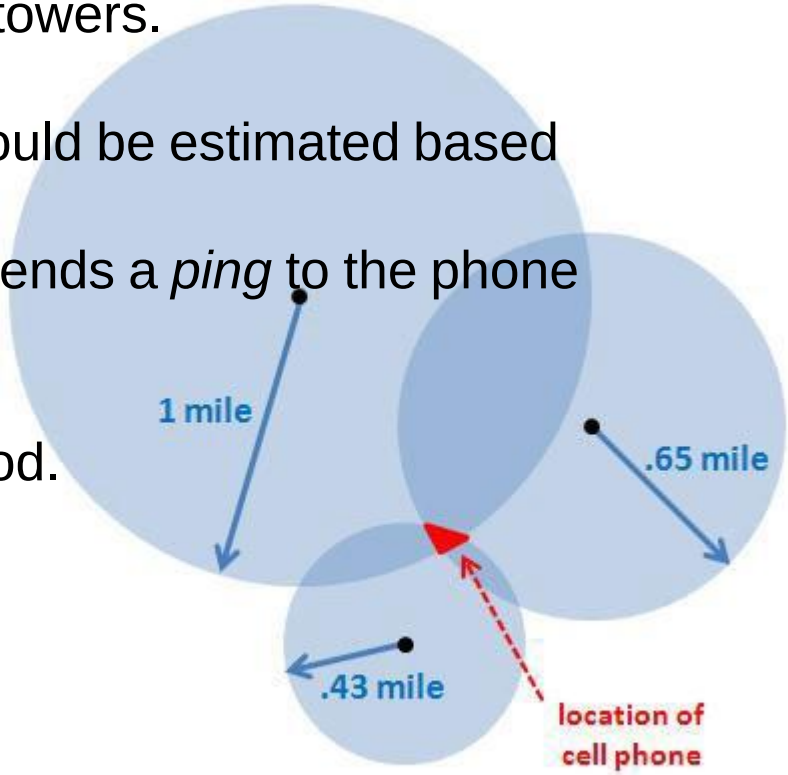
San Jose in Costa Rica

Cell Tower Triangulation

An alternative method to determine the location of a cell phone is to estimate its distance to three nearby cell towers.

Distance of the phone to each antenna could be estimated based upon the
lag time between the moment the tower sends a *ping* to the phone and receives the answering ping back.

Quite similar to the 2D-Trilateration Method.

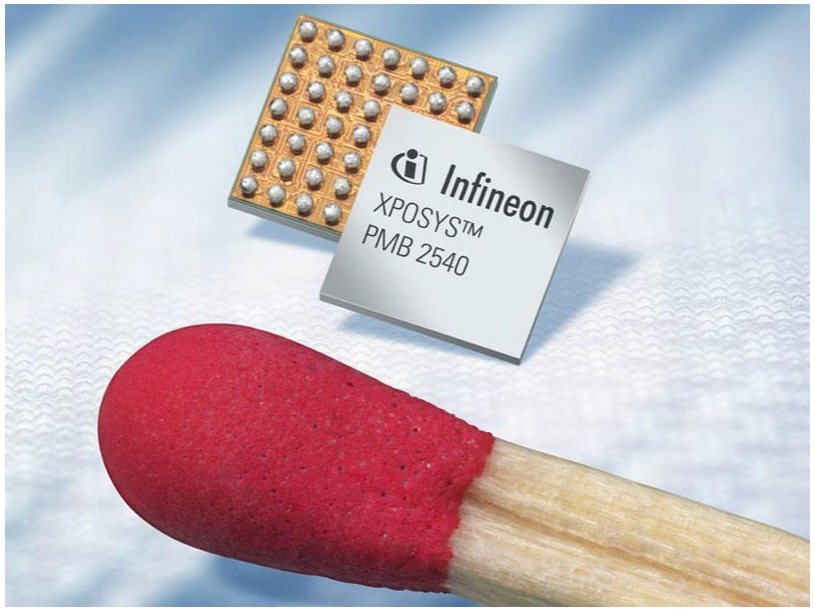


Triangulation - cell phone detected within a certain radius of each of 3 cell towers – the area where each cell tower overlaps the phone is where it is pinpointed.

Reference:
http://searchengineland.com/cell-phone-triangulation-accuracy-is-all-over-the-map-14790

Android Location Classes

The Android API provides Location data based on a variety of methods including: *Cell Tower Triangulation*, and most commonly *GPS chip readings*.



GPS is the most common location provider on the Android based phones.

It offers the most accuracy.

Picture: Epson Infineon GPS (2.8 x 2.9mm)

Reference: http://gizmodo.com/5152146/

Android Location Classes

Address	A class representing an Address, i.e, a set of strings describing a location.
Criteria	A class indicating the application criteria for selecting a location provider.
Geocoder	A class for handling geocoding.
GpsSatellite	This class represents the current state of a GPS satellite.
GpsStatus	This class represents the current state of the GPS engine.
Location	A class representing a geographic location sensed at a particular time (a "fix").
LocationManager	This class provides access to the system location services.
LocationProvider	An abstract superclass for location providers
GpsStatus.Listener	Used for receiving notifications when GPS status has changed.
GpsStatus.NmeaListener	Used for receiving NMEA sentences from the GPS.
LocationListener	Used for receiving notifications from the *LocationManager* when the location has changed.

Location Class

- A class representing a geographic location sensed at a particular time.
- A location consists of a latitude and longitude, a UTC timestamp and *optionally* information on altitude, speed, and bearing.
- Information specific to a particular provider or class of providers may be communicated to the application using **getExtras**, which returns a Bundle of *key/value* pairs.
- Each provider will only provide those entries for which information is available.

CONSTANTS	
Location.FORMAT_DEGREES	Constant used to specify formatting of a latitude or longitude in the form [+-]DDD.DDDDD where D indicates degrees.
Location.FORMAT_MINUTES	Constant used to specify formatting of a latitude or longitude in the form "[+-]DDD:MM.MMMMM" where D indicates degrees and M indicates minutes of arc (1 minute = 1/60th of a degree).
Location.FORMAT_SECONDS	Constant used to specify formatting of a latitude or longitude in the form "[+-] DDD:MM:SS.SSSSS" where D indicates degrees, M indicates minutes of arc, and S indicates seconds of arc (1 minute = 1/60th of a degree, 1 second = 1/3600th of a degree).

Location Values Format

The three common formats:

DDD° MM' SS.S"	Degrees, Minutes and Seconds
DDD° MM.MMM'	Degrees and Decimal Minutes
DDD.DDDDD°	Decimal Degrees

There are sixty seconds in a minute (60" = 1') and
There are sixty minutes in a degree (60' = 1°).

Examples:

DDD° MM' SS.S”	32° 18' 23.1" N	122° 36' 52.5" W
DDD° MM.MMM’	32° 18.385' N	122° 36.875' W
DDD.DDDDD°	32.30642° N	122.61458° W
or	+32.30642,	-122.61458

Location Manager

This class provides access to the system location services.

These services allow applications

1.To *obtain periodic updates of the device's geographical location*,

2.or to fire an application-specified **Intent** when the *device enters the proximity of a given geographical location*.

```
String service_name = Context.LOCATION_SERVICE;
LocationManager locationManager = (LocationManager) getSystemService(service_name)
```

Location Manager's Methods

void	addProximityAlert (double latitude, double longitude, float radius, long expiration, PendingIntent intent) Sets a proximity alert for the location given by the position (latitude, longitude) and the given radius.
String	getBestProvider (Criteria criteria, boolean enabledOnly) Returns the name of the provider that best meets the given criteria.
GpsStatus	getGpsStatus (GpsStatus status) Retrieves information about the current status of the GPS engine.
Location	getLastKnownLocation (String provider) Returns a Location indicating the data from the last known location fix obtained from the given provider.
LocationProvider	getProvider (String name) Returns information associated with the location provider of the given name, or null if no provider exists by that name.
List<String>	getProviders (Criteria criteria, boolean enabledOnly) Returns a list of the names of LocationProviders that satisfy the given criteria, or null if none do.
void	requestLocationUpdates (String provider, long minTime, float minDistance, PendingIntent intent) Registers the current activity to be notified periodically by the named provider.
void	requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener) Registers the current activity to be notified periodically by the named provider.
void	setTestProviderStatus (String provider, int status, Bundle extras, long updateTime) Sets mock status values for the given provider.

LocationProvider Class

- An *abstract superclass* for location providers.
- A location provider *supplies periodic reports on the geographical location of the device.*
- Each provider has a set of criteria under which it may be used; for example,
  - some providers require GPS hardware and visibility to a number of satellites;
  - others require the use of the cellular radio,
  - or access to a specific carrier's network,
  - or access to the internet.
- They may also have *different battery consumption* characteristics or *monetary costs* to the user.
- The **Criteria** class allows providers to be selected based on user-specified criteria.

LocationProvider's Methods

Public Methods	
abstract int	getAccuracy() Returns a constant describing horizontal accuracy of this provider.
String	getName() Returns the name of this provider.
abstract int	getPowerRequirement() Returns the power requirement for this provider.
abstract boolean	hasMonetaryCost() true if the use of this provider may result in a monetary charge to the user, false if use is free.
boolean	meetsCriteria(Criteria criteria) Returns true if this provider meets the given criteria, false otherwise.
abstract boolean	requiresCell() true access to a cellular network (to make use of cell tower IDs) is needed, false otherwise.
abstract boolean	requiresNetwork() true if the provider requires access to a data network (e.g., the Internet), false otherwise.
abstract boolean	requiresSatellite() true if access to a satellite-based positioning system (e.g., GPS) is needed, false otherwise.
abstract boolean	supportsAltitude() Returns true if the provider is able to provide altitude information, false otherwise.
abstract boolean	supportsBearing() Returns true if the provider is able to provide bearing information, false otherwise.
abstract boolean	supportsSpeed() Returns true if the provider is able to provide speed information, false otherwise.

LocationProvider Class

- Provider Reference

```
String providerName = LocationManager.GPS_PROVIDER;
LocationProvider gpsProvider;
gpsProvider = locationManager.getProvider(providerName);
```

- Common Location Providers:
  - LocationManager.GPS_PROVIDER
  - LocationManager.NETWORK_PROVIDER

- Getting list of all providers

```
boolean enabledOnly = true;
List<String> providers = locationManager.getProviders(enabledOnly);
```

Finding Location Providers Using Criteria

- Provider with specific requirements

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false); criteria.setBearingRequired(false);
criteria.setSpeedRequired(false); criteria.setCostAllowed(true);

String bestProvider = locationManager.getBestProvider(criteria, true);
```

- To get all matching Providers

```
List<String> matchingProviders = locationManager.getProviders(criteria, false);
```

LocationListener Class

Used for receiving notifications from the **LocationManager** when the *location has changed*.

These methods are called if the **LocationListener** has been *registered* with the location manager service using the method:

```
requestLocationUpdates (Provider, minTime, minDistance, LocationListener)
```

LocationListener's Methods

abstract void	onLocationChanged (Location location) Called when the location has changed.
abstract void	onProviderDisabled (String provider) Called when the provider is disabled by the user.
abstract void	onProviderEnabled (String provider) Called when the provider is enabled by the user.
abstract void	onStatusChanged (String provider, int status, Bundle extras) Called when the provider status changes.

LocationListener

```
String provider = LocationManager.GPS_PROVIDER; int t = 5000; // milliseconds
int distance = 5; // meters

LocationListener myLocationListener = new LocationListener() { public void
onLocationChanged(Location location) {
// Update application based on new location.
}
public void onProviderDisabled(String provider){
// Update application if provider disabled.
}
public void onProviderEnabled(String provider){
// Update application if provider enabled.
}
public void onStatusChanged(String provider, int status,  Bundle extras){
// Update application if provider hardware status changed.
}
};

locationManager.requestLocationUpdates(provider, t, distance, myLocationListener);
```

FINDING YOUR LOCATION

- Reference Location Manager

```
String service_name = Context.LOCATION_SERVICE;
LocationManager locationManager = (LocationManager) getSystemService(service_name)
```

- Permissions in Manifest

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

- Last location “fix”

```
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
```

Example – Obtain Location from GPS

- In this example we request **GPS** services and display *latitude* and *longitude* values on the UI.
- Notes
  1. Observe the *GPS chip is not a synchronous device* that will immediately respond to a "*give me a GPS reading*" call.
  2. In order to engineer a good solution that takes into account the potential delays in obtaining location data we place the UI in the main activity and the request for location call in a background service.
  3. Remember the service runs in the same process space as the main activity, therefore for the sake of responsiveness we must place the logic for location data request in a separate parallel thread.

Geocoding

Geocoding

- Geocoding lets you translate between street addresses and longitude/ latitude map coordinates.
- The geocoding lookups are done on the server, so your applications will require you to include an Internet uses-permission in your manifest, as shown here:

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

- The Geocoder class provides access to two geocoding functions:

➢ **Forward geocoding** Finds the latitude and longitude of an address

➢ **Reverse geocoding** Finds the street address for a given latitude and longitude

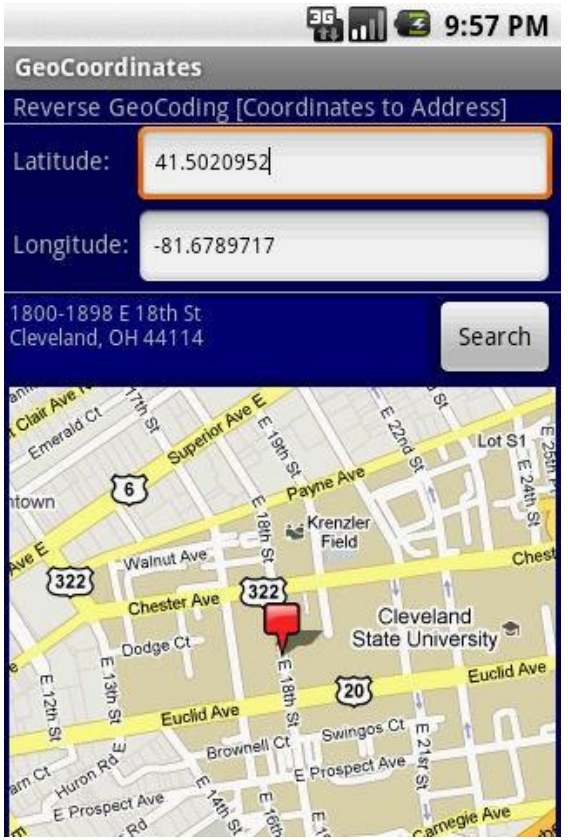
For more details:
http://developer.android.com/reference/android/location/Geocoder.html
http://developer.android.com/reference/android/location/Address.html

Reverse Geocoding

```
Geocoder gc= new Geocoder(context, Locale.US);
List<Address> streets = gc.getFromLocation(latitude, longitude, 1);
```

```
location = locationManager.getLastKnownLocation
                        (LocationManager.GPS_PROVIDER);
double lat = location.getLatitude();
double long = location.getLongitude();

List<Address> addresses = null;
Geocoder gc = new Geocoder(this, Locale.getDefault());
try {
        addresses = gc.getFromLocation(lat,long, 10);
} catch (IOException e) {}
```



Forward Geocoding

```
Geocoder gc= new Geocoder(this);
// get decimal coordinates for up to 5 (best) matching locations
List<Address> lstFoundAddresses= gc.getFromLocationName(txtStreetAddress, 5);
```

```
Geocoder fwdGeocoder = new Geocoder(this, Locale.US);
String streetAddress = "160 Riverside Drive,
New York, New York";
List<Address> locations = null;
try {
locations =
fwdGeocoder.getFromLocationName(streetAddress, 10);
} catch (IOException e) {}
```

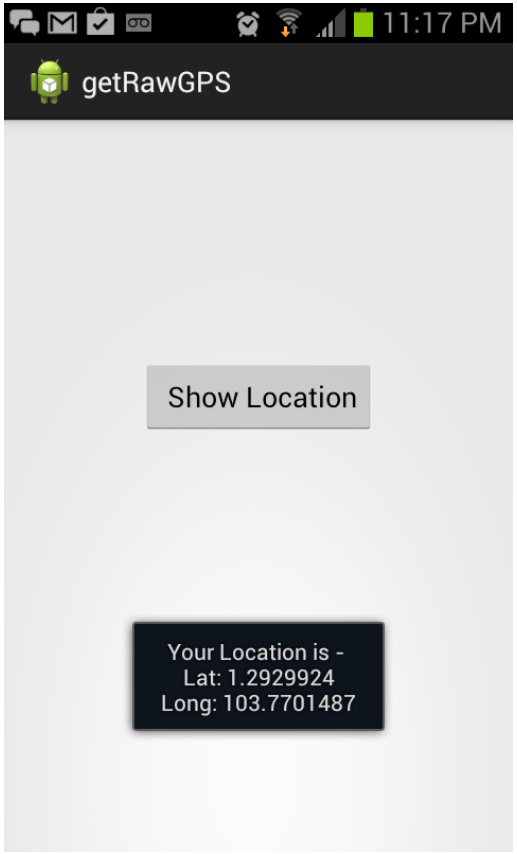
To Do

- Example 1: Obtain Location from GPS

Example 1

Example 1: Obtain Location from GPS

Layout




```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.getrawgps.MainActivity" >

    <Button
        android:id="@+id/btnShowLocation"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="134dp"
        android:text=" Show Location" />

</RelativeLayout>
```

Example 1: Obtain Location from GPS

Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.sjsu.android.example1">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".GPSTracker" />
    </application>

</manifest>
```

Example 1: Obtain Location from GPS

MainActivity

```
package edu.sjsu.android.example1;

import androidx.appcompat.app.AppCompatActivity;
import android.location.Location;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

Example 1: Obtain Location from GPS

MainActivity

```
public class MainActivity extends AppCompatActivity {
    Button btnShowLocation;
    GPSTracker gps;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnShowLocation = findViewById(R.id.btnShowLocation);
        // show location button click event
        btnShowLocation.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                // create class object
                gps = new GPSTracker(MainActivity.this);
                Location location = gps.getLocation();
                // check if GPS enabled
                if (gps.canGetLocation()) {
                    double latitude = location.getLatitude();
                    double longitude = location.getLongitude();
                    Toast.makeText(getApplicationContext(), "You Location is - \nLat "
                            + latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
```

Example 1: Obtain Location from GPS

GPSTracker class

```
package edu.sjsu.android.example1;

import android.Manifest;
import android.app.*;
import android.content.*;
import android.content.pm.PackageManager;
import android.location.*;
import android.os.*;
import android.provider.Settings;
import android.util.Log;

import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
```

Example 1: Obtain Location from GPS

GPSTracker Class

```
public class GPSTracker extends Service implements LocationListener {
    private final Context mContext;
    // flag for GPS status
    boolean isGPSEnabled = false;
    // flag for network status
    boolean isNetworkEnabled = false;
    boolean canGetLocation = false;
    Location location; // location
    double latitude; // latitude
    double longitude; // longitude
    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10; // 10 meters
    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60; // 1 minute
    // Declaring a Location Manager
    protected LocationManager locationManager;

    public GPSTracker(Context context) {
        this.mContext = context;
    }
```

Example 1: Obtain Location from GPS

GPSTracker Class

```
public Location getLocation() {
    try {
        locationManager = (LocationManager) mContext
                .getSystemService(LOCATION_SERVICE);
        // getting GPS status
        isGPSEnabled = locationManager
                .isProviderEnabled(LocationManager.GPS_PROVIDER);
        // getting network status
        isNetworkEnabled = locationManager
                .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        if (checkPermission() && (isNetworkEnabled || isGPSEnabled)) {
            this.canGetLocation = true;
            // First get location from Network Provider
            if (isNetworkEnabled) {
                locationManager.requestLocationUpdates(
                        LocationManager.NETWORK_PROVIDER,
                        MIN_TIME_BW_UPDATES,
                        MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                Log.d("Network", "Network");
                if (locationManager != null) {
                    location = locationManager
                            .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                    if (location != null) {
                        latitude = location.getLatitude();
                        longitude = location.getLongitude();
                    }
                }
            }
```

Example 1: Obtain Location from GPS

GPSTracker Class

```
            // if GPS Enabled get lat/long using GPS Services
            if (isGPSEnabled) {
                if (location == null) {
                    locationManager.requestLocationUpdates(
                            LocationManager.GPS_PROVIDER,
                            MIN_TIME_BW_UPDATES,
                            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    Log.d("GPS Enabled", "GPS Enabled");
                    if (locationManager != null) {
                        location = locationManager
                                .getLastKnownLocation(LocationManager.GPS_PROVIDER);
                        if (location != null) {
                            latitude = location.getLatitude();
                            longitude = location.getLongitude();
                        }
                    }
                }
            } else if(!checkPermission()){
                canGetLocation = false;
                requestPermission();
            } else if(!isGPSEnabled && !isNetworkEnabled){
                canGetLocation = false;
                showSettingAlert();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return location;
    }
```

Example 1: Obtain Location from GPS

GPSTracker Class

```
public boolean canGetLocation() {
    return this.canGetLocation;
}

 /**
  * Stop using GPS listener
  * Calling this function will stop using GPS in your app
  * */
 public void stopUsingGPS(){
     if(locationManager != null){
         locationManager.removeUpdates(GPSTracker.this);
     }
 }
```

```
private boolean checkPermission() {
    int result1 = ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_FINE_LOCATION);
    int result2 = ActivityCompat.checkSelfPermission(mContext, Manifest.permission.ACCESS_COARSE_LOCATION);
    return result1 == PackageManager.PERMISSION_GRANTED && result2 == PackageManager.PERMISSION_GRANTED;
}

private void requestPermission() {
    ActivityCompat.requestPermissions((Activity) mContext,
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION}, 100);
}
```

Example 1: Obtain Location from GPS

GPSTracker Class

```
/**
 * Function to show settings alert dialog
 * */
public void showSettingsAlert(){
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);
    // Setting Dialog Title
    alertDialog.setTitle("GPS is settings");
    // Setting Dialog Message
    alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");
    // Setting Icon to Dialog
    //alertDialog.setIcon(R.drawable.delete);
    // On pressing Settings button
    alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog,int which) {
            Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
            mContext.startActivity(intent);
        }
    });

    // on pressing cancel button
    alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
        dialog.cancel();
        }
    });

    // Showing Alert Message
    alertDialog.show();
}
```