CKR40 CS6313 Services and Mobile Middleware
Semester 2 2016/2017

LBS Middleware Service Final Report

Lecturer: Dr. Dan Grigoras

Student No.: 116220368
Student Name: Jingguo Jiang

1. Introduction

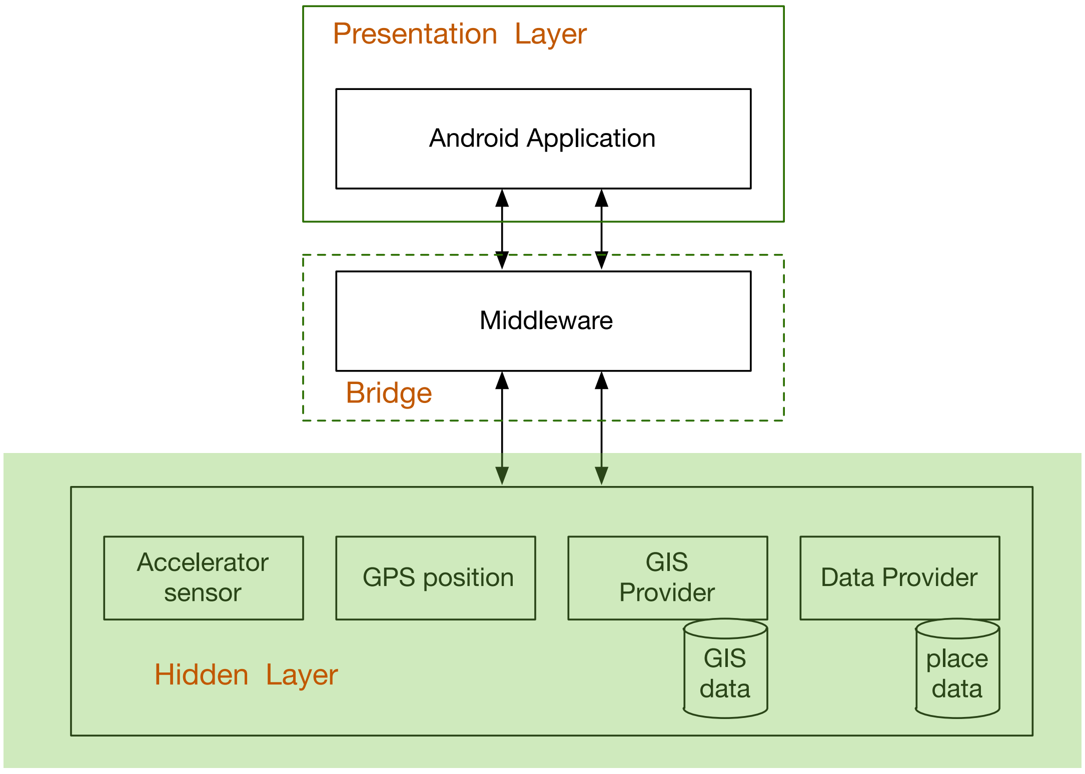
Location-based service (LBS) is based on location information which is a software-level service that uses location data to control and present features. As a category of middleware service, LBS's implementation is outcome of the integration of mobile network, GPS sensor and providers of data resources (e.g. Google Map, Bing, FourSquare). A lot of LBS application can be seen, nowadays, in many fields such as business, medicine, etc.

The present project is to implement a LBS in an Android application to help users find nearby places and display them in Google Map. Furthermore, a user can obtain the route of destination by virtue of Google Map application on mobile.

2. Service Architecture

In this project, two service providers: Google Map and Foursquare are employed to provide LBS service. Google Map server, as a GIS provider, can be used to determine the exact address. As we know, Android has an in-built GPS which can be used to determine the current location of a mobile device. The location will be represented by accurate latitude and longitude as a geolocation. After obtaining the geolocation, a HTTP request needs to be sent to Google server in order to reverse the geolocation into a real address. When a user shakes the mobile, it will trigger places which can be displayed in the screen. Android accelerator sensor will detect the shaking and communicate with data provider (FourSquare) to access place data and return them. Here, another HTTP request will be sent to FourSquare server.

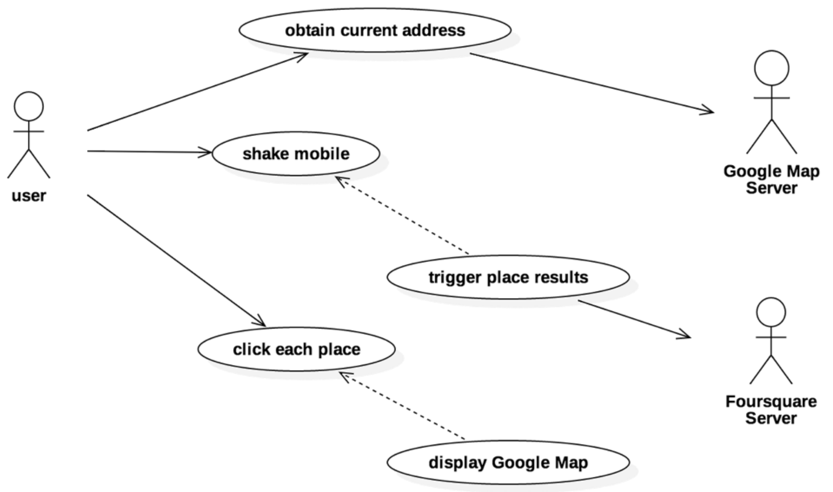
In the entire service ecosystem, middleware service acts as a bridge to link application (presentation layer) and service provider (hidden layer), as shown in figure below. The middleware gives the access to core functionalities including accelerator sensor, GPS position, GIS provider as well as data provider. Mobile middleware provides abundant APIs to support LBS implementation. For instance, Sensor/ SensorManager classes can be used for shaking feature implementation. LocationManager is a key class supporting positioning; LocationListener interface can be used to detect location change of a mobile device. HttpURLConnection class will be used to create network connection in order to obtain GIS and places data from Google Maps and Foursquare Server respectively.



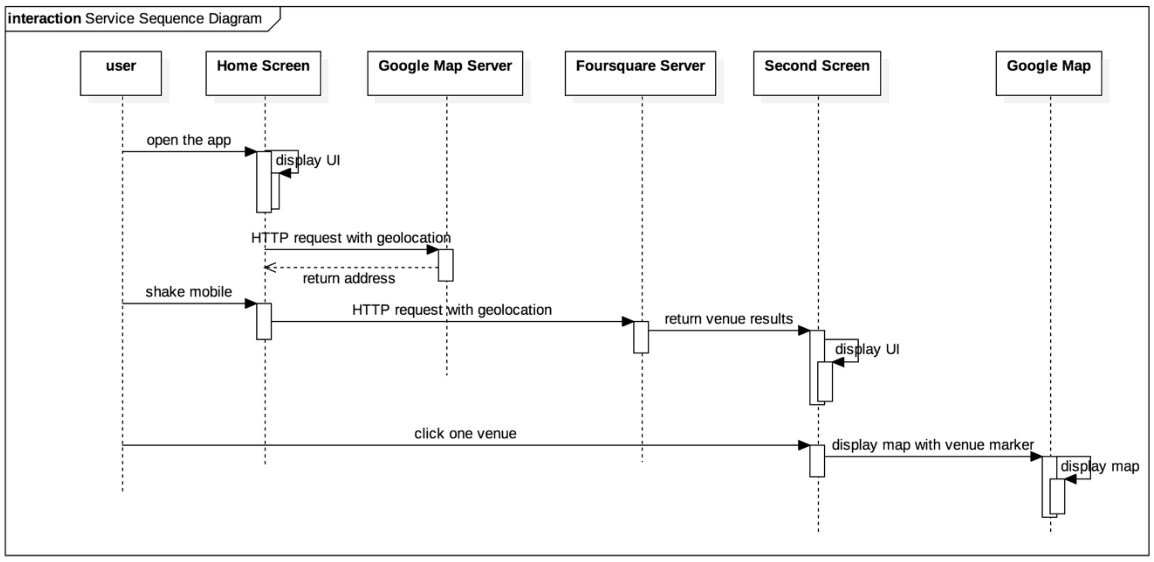
3. Service Design

The service architecture, as discussed before, makes it possible to develop an Android application. Before coding, it is necessary to perform system design (part of software engineering) in order to guarantee quality of the application development.

Use case modelling is to depict the interactions between the system and actors. It can be used to represent the system function. The application function can be seen in the use case diagram below. In the entire service system, three actors can be identified. ‘user’ actor represents the user of service through interacting with this application. ‘Google map server’ represents a GIS provider. ‘Foursquare server’ is regarded as place resources provider. ‘user’ opens the application, knows where he or she is, and gets nearby places. Each use case represents the interaction between a user and the application or the interaction between the application and the service providers (Google map server and Foursquare server).

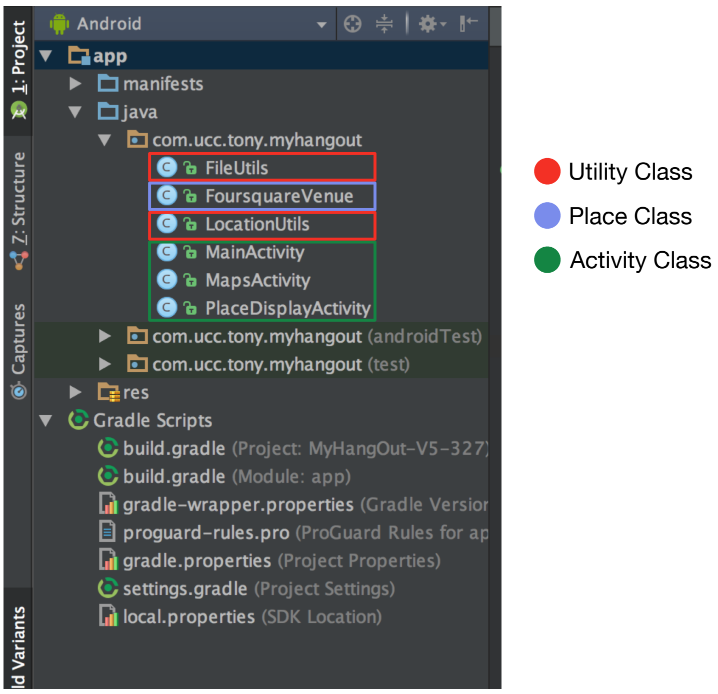


A Sequence Diagram models the collaboration of the objects based on the time sequence. It presents how the objects interact with others in a particular scenario of a use case. The following figure gives the overview of LBS service interaction with objects including user, interface objects as well as service providers.



4. Implementation

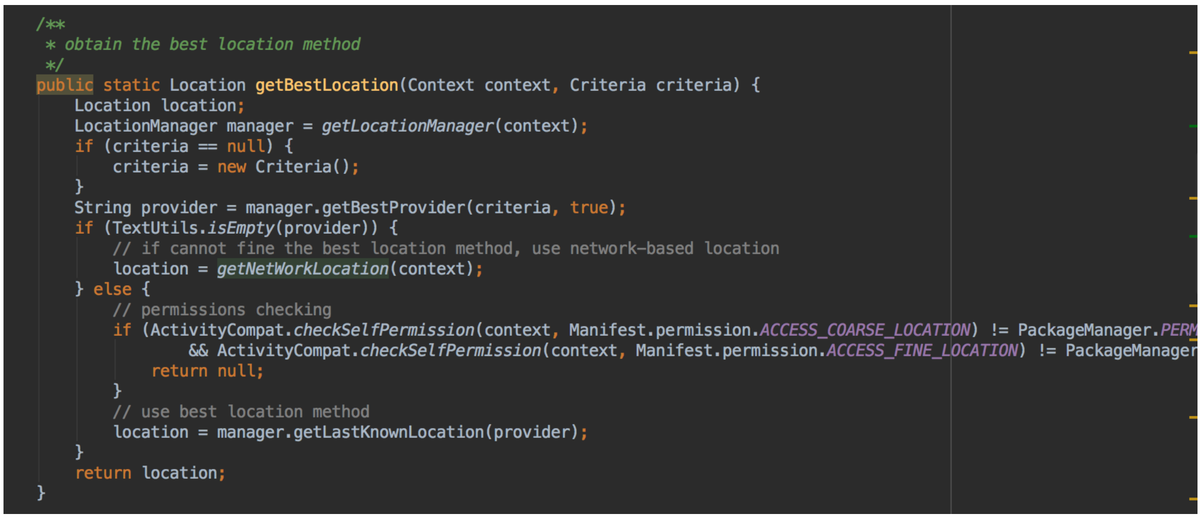
As shown below, I used Android Studio to develop this application. The application structure includes six classes. FileUtilis and LocationUtils are two utility classes. MainActivity, MapsActivity, and PlaceDisplayActivity are three Activity classes used to display interface elements and implement the action logic. FoursquareVenue class is used to encapsulate data which is accessed through Foursquare API.



```
Android
▼ app
  ▶ manifests
  ▼ java
    ▼ com.ucc.tony.myhangout
        FileUtils
        FoursquareVenue
        LocationUtils
        MainActivity
        MapsActivity
        PlaceDisplayActivity
    ▶ com.ucc.tony.myhangout (androidTest)
    ▶ com.ucc.tony.myhangout (test)
  ▶ res
▼ Gradle Scripts
    build.gradle (Project: MyHangOut-V5-327)
    build.gradle (Module: app)
    gradle-wrapper.properties (Gradle Versior
    proguard-rules.pro (ProGuard Rules for ap
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

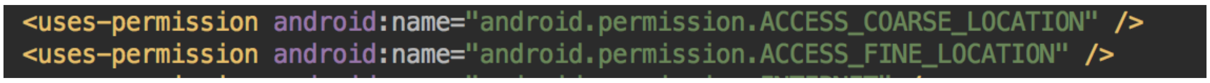
🔴 Utility Class
🔵 Place Class
🟢 Activity Class

LocationUtils class can facilitate positioning functionality. In this class, three types of methods which can perform positioning. There methods cover getGPSLocation(), getNetworkLocation, as well as getBestLocation(). Here, getBestLocation() method aims to choose best location approach in terms of Network conditions. To obtain the geolocation, I mainly called getBestLocation() in MainActivity.class.



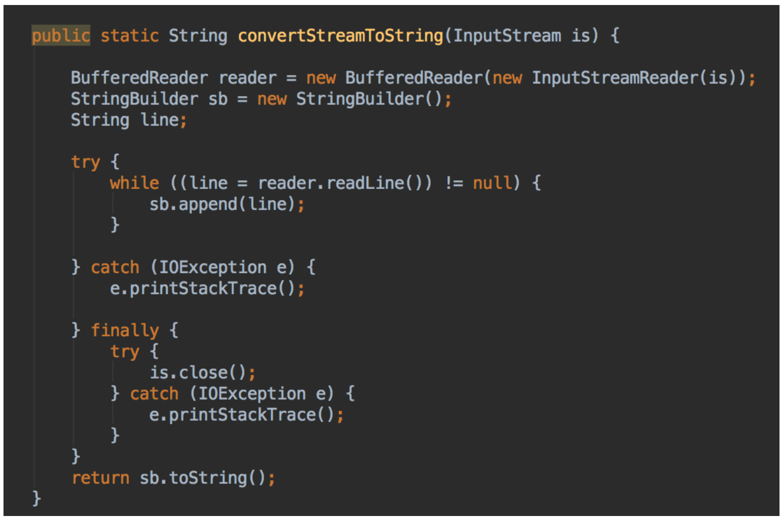
```java
/**
 * obtain the best location method
 */
public static Location getBestLocation(Context context, Criteria criteria) {
    Location location;
    LocationManager manager = getLocationManager(context);
    if (criteria == null) {
        criteria = new Criteria();
    }
    String provider = manager.getBestProvider(criteria, true);
    if (TextUtils.isEmpty(provider)) {
        // if cannot fine the best location method, use network-based location
        location = getNetWorkLocation(context);
    } else {
        // permissions checking
        if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERM
                && ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager
            return null;
        }
        // use best location method
        location = manager.getLastKnownLocation(provider);
    }
    return location;
}
```

Before using GPS function, I need to register permission in Manifest file, as shown in figure below.



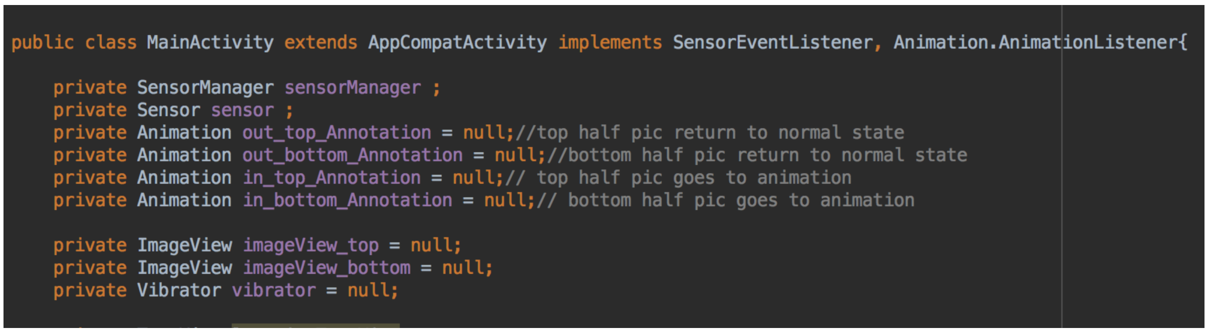
```xml
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

FileUtilis is used to read data from JSON file because a static method to achieve this, as shown below. I encapsulated a method in order to convert InputStream to String type.



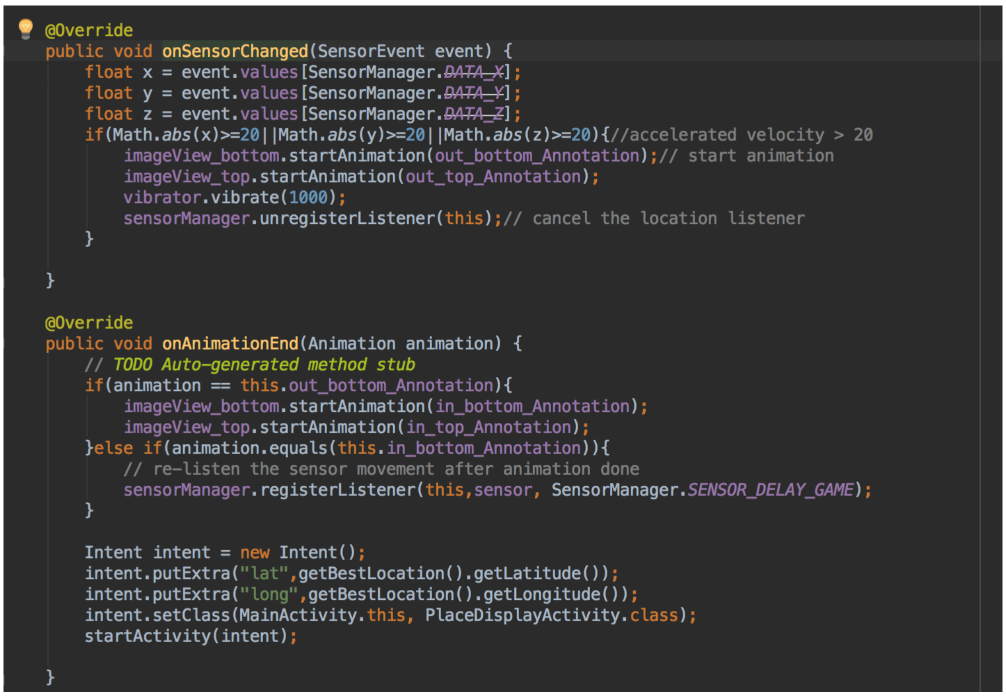
```java
public static String convertStreamToString(InputStream is) {

    BufferedReader reader = new BufferedReader(new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
    String line;

    try {
        while ((line = reader.readLine()) != null) {
            sb.append(line);
        }

    } catch (IOException e) {
        e.printStackTrace();

    } finally {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return sb.toString();
}
```

MainActivity is the application’s entry point; in this activity, shaking feature and Google Map service access are implemented. To develop shaking feature, MainActivity need to implement *SensorEventListener* and *AnimationListener.*



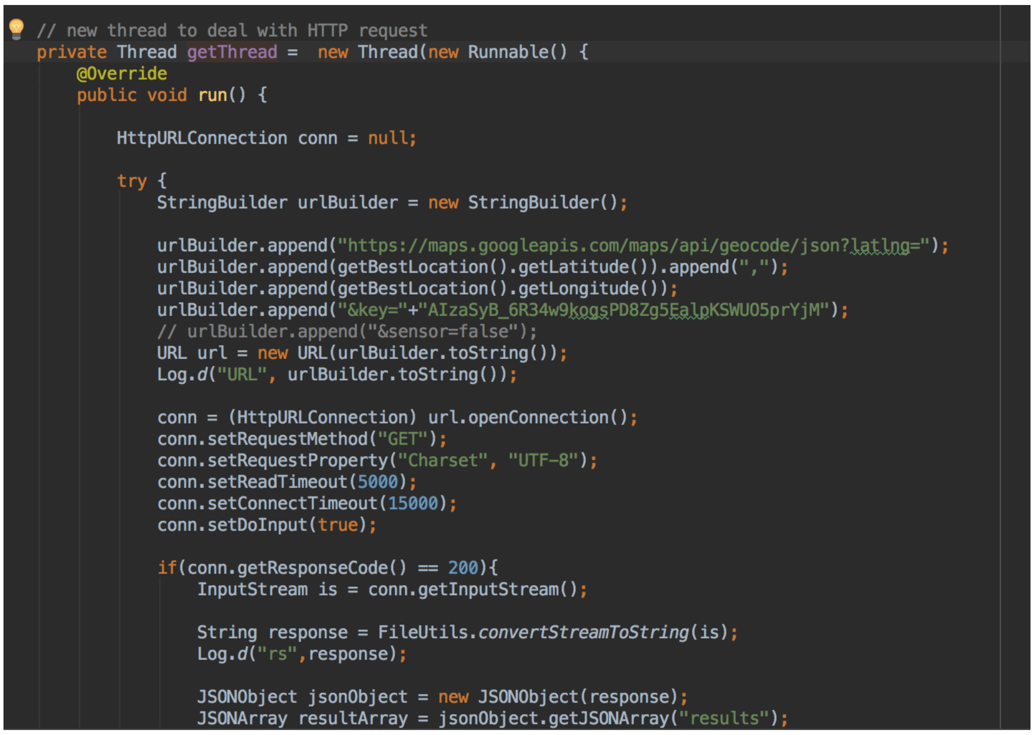
```java
public class MainActivity extends AppCompatActivity implements SensorEventListener, Animation.AnimationListener{

    private SensorManager sensorManager ;
    private Sensor sensor ;
    private Animation out_top_Annotation = null;//top half pic return to normal state
    private Animation out_bottom_Annotation = null;//bottom half pic return to normal state
    private Animation in_top_Annotation = null;// top half pic goes to animation
    private Animation in_bottom_Annotation = null;// bottom half pic goes to animation

    private ImageView imageView_top = null;
    private ImageView imageView_bottom = null;
    private Vibrator vibrator = null;
```

Then, the methods of these two interfaces are overwritten.



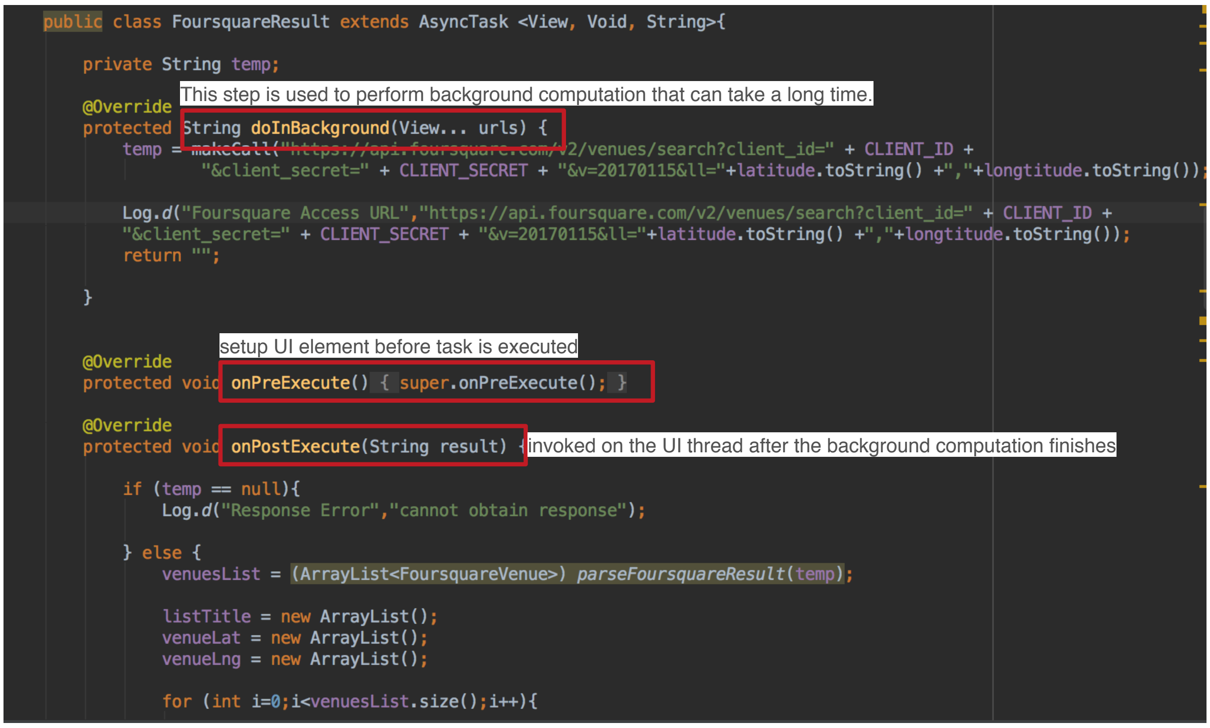
```java
@Override
public void onSensorChanged(SensorEvent event) {
    float x = event.values[SensorManager.~~DATA_X~~];
    float y = event.values[SensorManager.~~DATA_Y~~];
    float z = event.values[SensorManager.~~DATA_Z~~];
    if(Math.abs(x)>=20||Math.abs(y)>=20||Math.abs(z)>=20){//accelerated velocity > 20
        imageView_bottom.startAnimation(out_bottom_Annotation);// start animation
        imageView_top.startAnimation(out_top_Annotation);
        vibrator.vibrate(1000);
        sensorManager.unregisterListener(this);// cancel the location listener
    }

}

@Override
public void onAnimationEnd(Animation animation) {
    // TODO Auto-generated method stub
    if(animation == this.out_bottom_Annotation){
        imageView_bottom.startAnimation(in_bottom_Annotation);
        imageView_top.startAnimation(in_top_Annotation);
    }else if(animation.equals(this.in_bottom_Annotation)){
        // re-listen the sensor movement after animation done
        sensorManager.registerListener(this,sensor, SensorManager.SENSOR_DELAY_GAME);
    }

    Intent intent = new Intent();
    intent.putExtra("lat",getBestLocation().getLatitude());
    intent.putExtra("long",getBestLocation().getLongitude());
    intent.setClass(MainActivity.this, PlaceDisplayActivity.class);
    startActivity(intent);

}
```

For address access from Google Map server, a HTTP request is performed as below.



```java
// new thread to deal with HTTP request
private Thread getThread =  new Thread(new Runnable() {
    @Override
    public void run() {

        HttpURLConnection conn = null;

        try {
            StringBuilder urlBuilder = new StringBuilder();

            urlBuilder.append("https://maps.googleapis.com/maps/api/geocode/json?latlng=");
            urlBuilder.append(getBestLocation().getLatitude()).append(",");
            urlBuilder.append(getBestLocation().getLongitude());
            urlBuilder.append("&key="+"AIzaSyB_6R34w9kogsPD8Zg5EalpKSWUO5prYjM");
            // urlBuilder.append("&sensor=false");
            URL url = new URL(urlBuilder.toString());
            Log.d("URL", urlBuilder.toString());

            conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setRequestProperty("Charset", "UTF-8");
            conn.setReadTimeout(5000);
            conn.setConnectTimeout(15000);
            conn.setDoInput(true);

            if(conn.getResponseCode() == 200){
                InputStream is = conn.getInputStream();

                String response = FileUtils.convertStreamToString(is);
                Log.d("rs",response);

                JSONObject jsonObject = new JSONObject(response);
                JSONArray resultArray = jsonObject.getJSONArray("results");
```

In PlaceDisplayActivity, I employed another method to handle HTTP request. Instead of creating a new thread, I used *AsyncTask* class to set up tasks of the network access. AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.



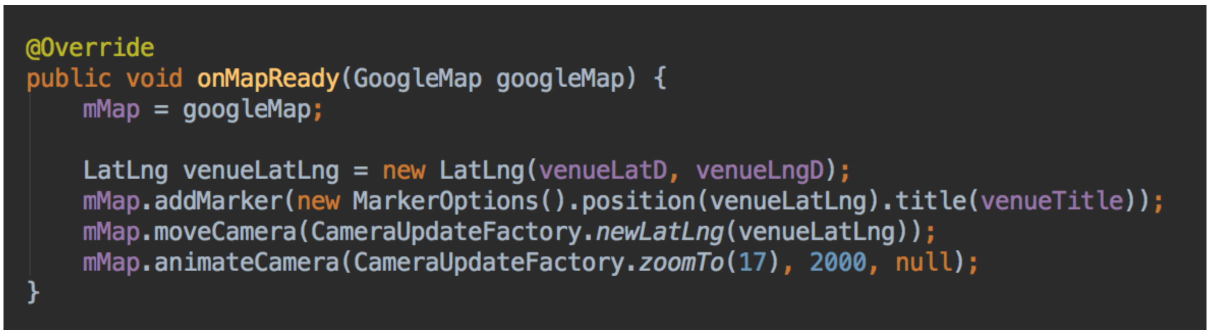
This step is used to perform background computation that can take a long time.

setup UI element before task is executed

invoked on the UI thread after the background computation finishes

```java
public class FoursquareResult extends AsyncTask <View, Void, String>{

    private String temp;

    @Override
    protected String doInBackground(View... urls) {
        temp = makeCall("https://api.foursquare.com/v2/venues/search?client_id=" + CLIENT_ID +
                "&client_secret=" + CLIENT_SECRET + "&v=20170115&ll="+latitude.toString() +","+longtitude.toString());

        Log.d("Foursquare Access URL","https://api.foursquare.com/v2/venues/search?client_id=" + CLIENT_ID +
        "&client_secret=" + CLIENT_SECRET + "&v=20170115&ll="+latitude.toString() +","+longtitude.toString());
        return "";

    }


    @Override
    protected void onPreExecute() { super.onPreExecute(); }

    @Override
    protected void onPostExecute(String result) {

        if (temp == null){
            Log.d("Response Error","cannot obtain response");

        } else {
            venuesList = (ArrayList<FoursquareVenue>) parseFoursquareResult(temp);

            listTitle = new ArrayList();
            venueLat = new ArrayList();
            venueLng = new ArrayList();

            for (int i=0;i<venuesList.size();i++){
```

In MapsActivity, the Google Map will be displayed with corresponding marker of the venue. The key method in this activity is onMapReady; this method is used to set up Google Map with a marker, marker's title and zooming parameters of the map.

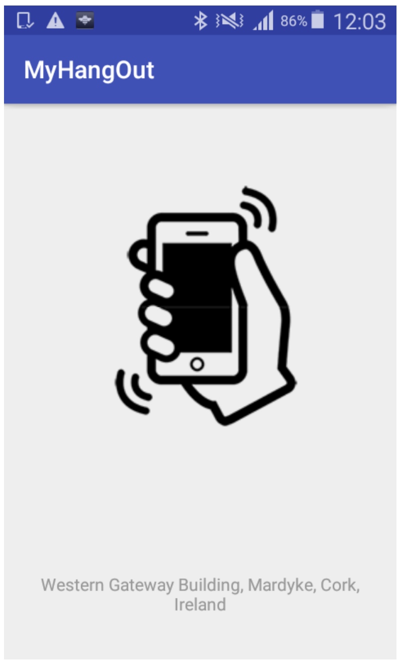


```java
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    LatLng venueLatLng = new LatLng(venueLatD, venueLngD);
    mMap.addMarker(new MarkerOptions().position(venueLatLng).title(venueTitle));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(venueLatLng));
    mMap.animateCamera(CameraUpdateFactory.zoomTo(17), 2000, null);
}
```

5. What I Learnt

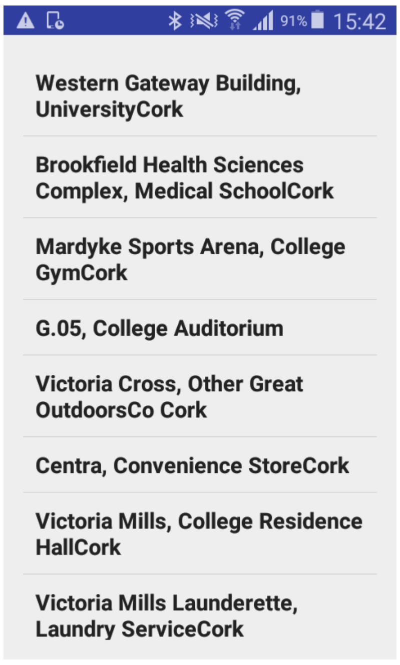
- How to read data from JSON file.
- Two approaches of handling HTTP requests.
- How to use Android GPS sensor and Accelerator sensor.

6. Final Demonstration



MyHangOut

Western Gateway Building, Mardyke, Cork, Ireland



Western Gateway Building, UniversityCork

Brookfield Health Sciences Complex, Medical SchoolCork

Mardyke Sports Arena, College GymCork

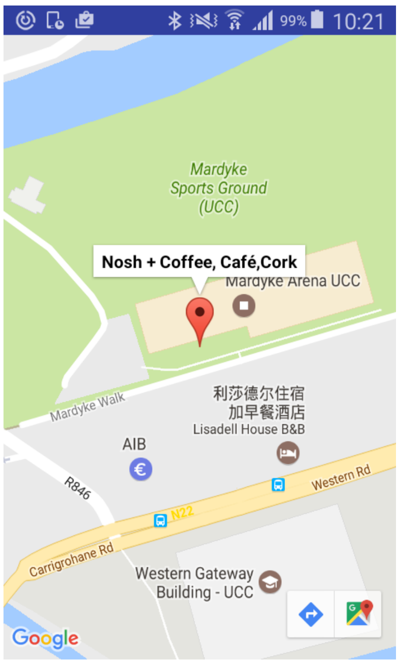
G.05, College Auditorium

Victoria Cross, Other Great OutdoorsCo Cork

Centra, Convenience StoreCork

Victoria Mills, College Residence HallCork

Victoria Mills Launderette, Laundry ServiceCork


Mardyke Sports Ground (UCC)
Nosh + Coffee, Café,Cork
Mardyke Arena UCC
Mardyke Walk
利莎德尔住宿加早餐酒店
Lisadell House B&B
AIB
R846
N22
Western Rd
Carrigrohane Rd
Western Gateway Building - UCC
Google